\((1 + \varepsilon)\)-Approximation for Facility Location in Data Streams

Artur Czumaj\(^*\) Christiane Lammersen\(^†\) Morteza Monemizadeh\(^‡\) Christian Sohler\(^§\)

Abstract
We consider the Euclidean facility location problem with uniform opening cost. In this problem, we are given a set of \(n\) points \(P \subseteq \mathbb{R}^2\) and an opening cost \(f \in \mathbb{R}^+\), and we want to find a set of facilities \(F \subseteq \mathbb{R}^2\) that minimizes

\[
f \cdot |F| + \sum_{p \in P} \min_{q \in F} d(p, q) ,
\]

where \(d(p, q)\) is the Euclidean distance between \(p\) and \(q\). We obtain two main results:

- A \((1 + \varepsilon)\)-approximation algorithm with running time
  \[
  \mathcal{O}(n \log n (\log n \log \log n + (\log \log n)^8 \varepsilon^{-2} \log(1/\varepsilon))) ,
  \]
  which is \(\mathcal{O}(n \log^2 n \log \log n)\) for any constant \(\varepsilon\).
- The first \((1 + \varepsilon)\)-approximation algorithm for the cost of the facility location problem for dynamic geometric data streams, i.e., when the stream consists of insert and delete operations of points from a discrete space \(\{1, \ldots, \Delta\}^2\). The streaming algorithm uses \(\big(\log \Delta\big)^2 \cdot \mathcal{O}(1)\) space.

Our PTAS is significantly faster than any previously known \((1 + \varepsilon)\)-approximation algorithm for the problem, and is also relatively simple. Our algorithm for dynamic geometric data streams is the first \((1 + \varepsilon)\)-approximation algorithm for the cost of the facility location problem with polylogarithmic space, and it resolves an open problem in the streaming area.

Both algorithms are based on a novel and simple decomposition of an input point set \(P\) into small subsets \(P_i\), such that:

- the cost of solving the facility location problem for each \(P_i\) is small (which means that for each \(P_i\), one needs to open only a small, polylogarithmic number of facilities),
- \(\sum_{i} \text{OPT}(P_i) \leq (1 + \varepsilon) \cdot \text{OPT}(P)\), where for a point set \(P\), \(\text{OPT}(P)\) denotes the cost of an optimal solution for \(P\).

1 Introduction
The facility location problem is one of the most extensively studied problems in combinatorial optimization and operations research (see, e.g., [7, 9]). The goal is for a given set of customers to find an optimal placement of facilities that minimizes the cost of customer service. Opening a facility at a certain location incurs a certain cost called opening cost, and serving customers at other locations incurs another type of cost called connection cost. The goal is to minimize the sum of both costs. Typical examples of applications include placement of servers in a network or location planning for stores, restaurants, etc. The focus of this paper is on the (uniform) facility location problem on the plane. Formally, the problem is defined as follows:

DEFINITION 1.1. (EUCLIDEAN UNIFORM FACILITY LOCATION PROBLEM) Let \(P = \{p_1, \ldots, p_n\}\) be a set of \(n\) points in the Euclidean plane \(\mathbb{R}^2\) and let \(f \in \mathbb{R}^+\) be some fixed parameter (called opening cost). The (uniform) facility location problem is to find a set \(F \subseteq \mathbb{R}^2\) of points (called open facilities) that minimizes

\[
\text{COST}(P, F) = |F| \cdot f + \sum_{p \in P} d(p, F) ,
\]

where \(d(p, F) = \min_{q \in F} d(p, q)\) is the Euclidean distance of \(p\) to the nearest open facility in \(F\). (The first term on the right-hand side of (1.1) is called the opening cost and the second one is the connection cost.)

Given a point set \(P \subseteq \mathbb{R}^2\), we denote the optimal cost of \(P\) by \(\text{OPT}_P\), and any set of optimal facilities by \(F_{\text{OPT}}\) (or \(\text{OPT}\) and \(F_{\text{OPT}}\), respectively, if \(P\) is clear from the context), i.e.,

\[
\text{OPT}_P = \min_{F \subseteq \mathbb{R}^2} \text{COST}(P, F) = \text{COST}(P, F_{\text{OPT}}) .
\]

We also consider the facility location problem for individual cells in \(\mathbb{R}^2\). For any cell \(c \in \mathbb{R}^2\), we define the cost of \(c\) as...
thermore, the partition is designed to enable to very quickly
\[ \text{COST}(\epsilon) = \min_{F \subseteq \mathbb{R}^2} \left\{ \sum_{p \in P \cap c} d(p, F) + |F| \cdot f \right\} , \]
where \( P \cap c \) denotes the set of all points in \( P \) that are contained in \( c \).

We will also consider the contribution of any individual cell in \( \mathbb{R}^2 \) for optimal facilities \( F_{\text{OPT}_P} \). Given a point set \( P \subseteq \mathbb{R}^2 \) and a cell \( c \subseteq \mathbb{R}^2 \), for a fixed set of optimal facilities \( F_{\text{OPT}_P} \), let \( \text{COST}_{\text{OPT}_P}(\epsilon) \) be the connection cost in \( \text{OPT}_P \) of all the points from \( P \) contained in \( c \) plus the opening cost in \( \text{OPT}_P \) for the open facilities contained in \( c \), i.e.,
\[ \text{COST}_{\text{OPT}_P}(\epsilon) = \sum_{p \in P \cap c} d(p, F_{\text{OPT}_P}) + |F_{\text{OPT}_P} \cap c| \cdot f . \]

We will study the facility location problem both in the classical and in the streaming model of computation for dynamic geometric data streams. Since the problem is known to be \( \text{NP} \)-hard [20], we will focus on \textit{approximation} algorithms. Furthermore, for the streaming model, given the structure of the problem (the output may be possibly of \( \Omega(|P|) \) size), we will only approximate the \textit{cost} of the facility location problem. Our motivation to consider the cost is

- in a dynamic scenario, for example, when maintaining a wireless network among moving robots, it is helpful to verify whether the current solution has still acceptable cost, and
- the solution computed by the facility location problem can also be viewed as a clustering problem with a regularization term for the number of cluster, i.e., one pays for each additional cluster a cost of \( f \) and thus the cost of the facility location problem can be viewed as a structural property of the input data.

### 1.1 Our contribution.

The main new technical contribution of this paper is a novel and simple partitioning scheme that allows to compute a good approximate solution to the facility location problem by taking the union of solutions for the partitioned input. That is, we can efficiently find a partition \( \Lambda \) of \( \Delta^2 \) such that \( \sum_{c \in \Lambda} \text{COST}(\epsilon) \leq (1 + \varepsilon) \cdot \text{OPT}_P \). Furthermore, the partition is designed to enable to very quickly compute almost optimal solutions for the cells \( c \in \Lambda \). For that, we use the observation that if \( \text{COST}(\epsilon) \) is small, then an \((1 + \varepsilon)\)-approximation to \( \text{COST}(\epsilon) \) can be found very quickly via a reduction to the \( k \)-median problem.

Using our new partitioning scheme, we develop two algorithms:

- a \((1 + \varepsilon)\)-approximation algorithm for the planar Euclidean facility location problem with uni-form cost running in time \( O(n \log^2 n \log \log n + n \log n (\log \log n)^8 2^{O((1/\varepsilon)/\varepsilon)}) \), and
- a \((1 + \varepsilon)\)-approximation algorithm for the \textit{cost} of the facility location problem in \textit{dynamic geometric data streams}, i.e., when the stream consists of insert and delete operations of points from the discrete space \( \{1, \ldots, \Delta\}^2 \), as introduced by Indyk [13]. The streaming algorithm uses \( (\log \Delta) \cdot O(1) \) space.

While polynomial-time approximation schemes for the planar Euclidean facility location problem with uniform cost have been known before [3, 17], our algorithm is arguably the simplest of those, and it is also significantly faster. No \((1 + \varepsilon)\)-approximation algorithm for the \textit{cost} of the facility location problem in \textit{dynamic geometric data streams} has been known before, and the existence of a \((1 + \varepsilon)\)-approximation algorithm with polylogarithmic space for this problem has been an open problem in the area of geometric streaming algorithms. The best approximation guarantee obtained in earlier works was constant [18]. Our streaming algorithm samples cells from the obtained partitioning scheme and approximates the cost by taking the average of the costs of the sample. The algorithm itself extends the ideas developed earlier in [1] and [14], with several new observations needed to overcome some technical obstacles caused by using these approaches to obtain a \((1 + \varepsilon)\)-approximation algorithm with polylogarithmic space. A byproduct of our work is an \( O(n \log n \log \log n) \)-time algorithm that computes a constant-factor approximation for the facility location problem; the running time of this algorithm improves upon earlier constant-time approximation algorithms.

### 1.2 Prior work.

The facility location problem has been extensively studied before in the combinatorial optimization and operations research. The problem is known to be \( \text{NP} \)-hard even in the geometric case. For general metrics, Hochbaum [12] gave the first non-trivial polynomial-time \( O(\log n) \)-approximation algorithm and Shmoys, Tur- dos and Aardal [22] gave the first constant-factor approximation algorithm for arbitrary metric spaces; the approximation factor has since been improved several times and the currently best polynomial-time approximation algorithm due to Li [19] achieves a 1.488-approximation. On the negative side, Guha and Khuller [10] showed that there is no polynomial-time \( \alpha \)-approximation algorithm for \( \alpha < 1.463 \), unless \( \text{NP} \subseteq \text{DTIME}(n^{O(\log \log n)}) \). The facility location problem has been also studied in the geometric setting. Arora, Raghavan, and Rao [3] gave the first PTAS for the geometric version of the problem, with a randomized algorithm achieving an \((1 + \varepsilon)\)-approximation in \( O(n^{1+O(1/\varepsilon)} \log n) \) time. This has been later improved by Kolliopoulos and Rao [17], who designed an \( O(\log^8 n 2^{O((1/\varepsilon)/\varepsilon)}) \)-time algo-
to the vector \((a, b)\).

We assume we have a unique numbering for cells in \(G_i\) so that when we refer to a cell \(c \in G_i\) we mean that this cell has a unique number \(c\) for \(1 \leq c \leq |G_i|\) according to this numbering. We call the four subcells of \(c \in G_i\), which are in the grid \(G_{i-1}\), the children of \(c\), and \(c\) is the parent cell of each one of these four subcells.

**Heavy and light cells.** In our construction, we will need a deterministic factor \(\alpha = O(1)\) approximation algorithm for the cost facility location problem \(\Lambda\). The algorithm will be fixed throughout the entire construction. (In specific applications, as we will use it later in the paper, for our PTAS, \(\Lambda\) will be the algorithm developed in this paper in Section 4, and for the streaming algorithm, we will assume that \(\Lambda\) is an algorithm that computes the cost of an optimal solution of the facility location problem.)

For any cell \(c\) in a grid \(G_i\), \(0 \leq i \leq \log \Delta + 1\), let us call \(c\) heavy if \(\Lambda\) returns that the cost of \(c\) is greater than \(\alpha n_f\); otherwise we call \(c\) light. We will require that \(n \geq c^* \cdot \left(\frac{\log \Delta + 1}{\varepsilon}\right)^2\), for a constant \(c^*\) defined later.

**Partition \(\Lambda\) of \([2\Delta]^2\).** We define a hierarchical partition \(\Lambda\) of \([2\Delta]^2\) into cells from a grid \(G_i\), \(0 \leq i \leq \log \Delta + 1\), to be the set of all cells \(c \in \bigcup_{i=0}^{\log \Delta} G_i\) that are light and whose all ancestors are heavy. More formally, we use a recursive procedure \(\text{Partition}(c)\), which we initially call with \(c\) being the entire square \([2\Delta]^2\):

\[
\text{Partition}(c) \rightarrow \text{Recursively finds a nested partition of cell } c
\]

- Consider cell \(c\) with four children \(c_1, c_2, c_3, c_4\).

- If cell \(c\) is heavy, then
  - recursively call \(\text{Partition}(c_1), \text{Partition}(c_2), \text{Partition}(c_3), \text{Partition}(c_4)\),
  - return the union of the obtained partitions as the partition of \(c\).

- Else, if \(c\) is light, then
  - return \{\(c\)\} itself as the partition of \(c\).

The procedure above returns a partition \(\Lambda\) of \([2\Delta]^2\) into a set of cells from grids \(G_i\), \(0 \leq i \leq \log \Delta + 1\).

Now we present our main theorem describing key properties of our partition \(\Lambda\).

**Theorem 2.1. (Partition Theorem)** Let \(\rho \in (0, 1)\) be any constant. Let \(P\) be any set of points in \([\Delta]^2\) that is randomly shifted in \([2\Delta]^2\). Then there are constants \(c = c(\rho), c^* = c^*(\rho)\) such that if \(n \geq c^* \cdot \left(\frac{\log \Delta + 1}{\varepsilon}\right)^2\) then partition \(\Lambda\) of \([2\Delta]^2\) satisfies the following properties:

---

A preliminary version of that paper [16] claimed a faster running time, but this has been later retracted and corrected in [17].
(1) for every cell \( c \in \Lambda \), \( \text{COST}(c) \leq \frac{c^2}{(\log \Delta + 2)^2} \cdot f \),

(2) \( \sum_{c \in \Lambda} \text{COST}(c) \leq (1 + \varepsilon) \cdot \text{OPT}_P \) with probability at least \( 1 - \omega \).

The rest of this section is devoted to the proof of Theorem 2.1. We first introduce useful notation in Section 2.1, then we prove basic properties of our construction in Sections 2.2 and 2.3, and we finalize the proof in Section 2.4.

In the entire analysis below we assume that \( F_{\text{OPT}, P} \) is a fixed optimal set of facilities, and with that, \( \text{COST}_{\text{OPT}, P}(c) \) is well defined for any cell \( c \).

---

**Figure 1:** (a) Partition of a cell \( c \) into a core and a frame, and (b) the buffer-net of \( I_c \).

---

**2.1 Frames, cores, and buffer-nets.** Let us fix \( \lambda = \frac{\varepsilon}{12(1+3/\varepsilon)(\log \Delta + 2)} \). A core of a cell \( c \) is the \((1 - 2\lambda)\ell_c \times (1 - 2\lambda)\ell_c \) square whose center is at the center of \( c \). A frame of a cell \( c \) is obtained from \( c \) by removing the core of \( c \). See Figure 1.a. (In other words, a frame consists of all points in the plane contained in \( c \) that are at distance less than \( \lambda \cdot \ell_c \) from the boundary of \( c \), and a core of \( c \) consists of all points in the plane contained in \( c \) that are at distance greater than or equal to \( \lambda \cdot \ell_c \) from the boundary of \( c \)). For a cell \( c \), let \( I_c \) denote its core and \( B_c \) denote its frame.

**Buffer-nets.** In our construction, we will frequently allow to open new facilities around either a core or a frame of a cell. Consider any cell \( c \) and its core \( I_c \) and frame \( B_c \). \( B_c \) may be seen as union of four stripes around \( I_c \) of size \( (\lambda \cdot \ell_c) \times \ell_c \) each (cf. Figure 1.a). Place \( 1/\lambda \) points in each stripe, by first partitioning each stripe into \( 1/\lambda \) disjoint \((\lambda \cdot \ell_c) \times (\lambda \cdot \ell_c)\) squares and then choosing the center of each square (cf. Figure 1.b). Call the obtained points the buffer-net of \( I_c \). Note that each buffer-net of a core has \( 4(\frac{1}{\lambda} - 1) \) points.

A similar construction can be done for \( B_c \). Let us draw eight stripes of width \( \lambda \cdot \ell_c \) around the boundary of \( B_c \): four \((\lambda \cdot \ell_c) \times ((1 + 2\lambda) \cdot \ell_c)\) stripes outside \( B_c \) and four \((\lambda \cdot \ell_c) \times ((1 - 2\lambda) \cdot \ell_c)\) stripes inside \( B_c \) (see Figure 2.a). (We call the union of the stripes the boundary of \( B_c \) and denote it by \( \mathcal{Q}_c^2 \)). Then, we partition each stripe into disjoint \((\lambda \cdot \ell_c) \times (\lambda \cdot \ell_c)\) squares and place points in the center of each square (see Figure 2.b). The obtained points are called the buffer-net of \( B_c \). Note that each buffer-net of a frame has \( 8(\frac{1}{\lambda} - 1) \) points.

The following key properties of buffer-nets will be used in the paper.

**Claim 2.1. (Buffer-net property for cores)** For any cell \( c \), any point \( p \in I_c \), and any point \( q \notin c \), there is a point \( x \) in the buffer-net of \( I_c \) such that

\[
d(p, x) \leq d(p, q)
\]

**Proof.** Let \( z \) be the minimum distance of \( p \) to the frame of \( c \). Since the frame of \( c \) has a width of \( \lambda \ell_c \) and due to the construction of the buffer-net of \( I_c \), the distance of any point from the border of the frame of \( c \) to the closest point in the buffer-net of \( I_c \) is at most \( \sqrt{2} \lambda \ell_c/2 \). Thus, by triangle inequality, the distance of \( p \) to its closest point \( x \) in the buffer-net of \( I_c \) is at most \( z + \sqrt{2} \lambda \ell_c/2 < z + \lambda \ell_c \). Since the frame of \( c \) has a width of \( \lambda \ell_c \), the distance of \( p \) to the closest point \( q \notin c \) is at least \( z + \lambda \ell_c \). \( \square \)

**Claim 2.2. (Buffer-net property for frames)** For any cell \( c \), any point \( p \in B_c \), and any point \( q \) that is not contained in \( B_c \) nor in the boundary of \( B_c \), there is a point \( x \) in the buffer-net of \( B_c \) with \( d(p, x) \leq d(p, q) \).

**Proof.** The proof is the same as that of Claim 2.1, with the only difference that \( p \in B_c \) and \( q \) is a point that is not contained in \( B_c \) nor in the boundary of \( B_c \) and \( z \) is the minimum distance of \( p \) to the boundary of \( B_c \). \( \square \)
2.2 Bounding the cost of cells. We will now prove three auxiliary lemmas bounding the cost of cores and frames.

**Lemma 2.1.** For every cell \( c \), we have

\[
\text{cost}(I_c) \leq \text{cost}_{\text{OPT},p}(c) + \frac{4f}{\lambda} .
\]

**Proof.** We will construct a feasible solution for \( I_c \) with the cost upper bounded by \( \text{cost}_{\text{OPT},p}(c) + 4\left(\frac{1}{k} - 1\right)f \). For this purpose, we open a facility at every point in \( F_{\text{opt},c} \cap c \). Furthermore, we open facilities for every point in the buffer net of \( I_c \). For every point \( p \in P \cap I_c \), if its nearest open facility in \( F_{\text{opt},c} \) is contained in \( c \), then we connect \( p \) to that facility. Otherwise, we connect \( p \) to the nearest point in the buffer net of \( I_c \). Claim 2.1 ensures that in the latter case, the chosen point \( q \) from the buffer-net of \( I_c \) satisfies \( d(p, q) \leq d(p, F_{\text{opt},c}) \).

We have constructed a feasible solution to the facility location for \( I_c \) whose cost is upper bounded by \( \sum_{p \in P \cap I_c} d(p, F_{\text{opt},c}) \) plus the cost of opening facilities in \( F_{\text{opt},c} \cap c \) plus the cost of opening some facilities in the buffer-net of \( I_c \). Since the number of points in the buffer-net of \( I_c \) is at most \( 4\left(\frac{1}{k} - 1\right) \) and since

\[
\text{cost}_{\text{OPT},p}(c) = \sum_{p \in P \cap I_c} d(p, F_{\text{opt},c}) + |F_{\text{opt},c} \cap c| \cdot f \\
\leq \sum_{p \in P \cap I_c} d(p, F_{\text{opt},c}) + |F_{\text{opt},c} \cap c| \cdot f ,
\]

we obtain the following:

\[
\text{cost}(I_c) \leq \sum_{p \in P \cap I_c} d(p, F_{\text{opt},c}) + |F_{\text{opt},c} \cap c| \cdot f + 4\left(\frac{1}{k} - 1\right)f \\
\leq \text{cost}_{\text{OPT},p}(c) + 4\left(\frac{1}{k} - 1\right)f .
\]

**Lemma 2.2.** For every cell \( c \), we have

\[
\text{cost}(B_c) \leq \text{cost}_{\text{OPT},p}(B_c \cup Q_c) + \frac{8f}{\lambda} ,
\]

and

\[
\text{cost}(c) \leq \text{cost}_{\text{OPT},p}(c \cup Q_c) + \frac{8f}{\lambda} .
\]

**Proof.** The proof is the same as that of Lemma 2.1, with the only difference in a different number of points in the buffer net of \( B_c \), which is now \( 8\left(\frac{1}{k} - 1\right) \) (versus \( 4\left(\frac{1}{k} - 1\right) \)) in the proof of Lemma 2.1), and in the use of Claims 2.1 and 2.2 instead of solely relying on Claim 2.1.

**Lemma 2.3. For our partition \( \Lambda \), the following bound holds:**

\[
\mathbb{E}\left[ \sum_{c \in \Lambda} \text{cost}_{\text{OPT},p}(B_c \cup Q_c) \right] \leq 12\lambda (\log \Delta + 2) \cdot \text{OPT}_{\text{P}} ,
\]

where the expectation is over the choice of the random choice of the shift of the dissection.

**Proof.** Let us take the nested grids \( G_0, G_1, \ldots, G_{\log \Delta+1} \), and all cells in each \( G_i \). For each cell, let us take its frame and let \( \mathcal{W}_i \) be the union of frames and their boundaries for all cells in \( G_i \). It is easy to see that the area of each \( \mathcal{W}_i \) is at most \( 12\lambda \) fraction of the total area (the area of all frames is at most \( 4\lambda \) fraction of the total area and the area of all boundaries of the frames is at most \( 8\lambda \) fraction of the total area). Hence, for any point \( p \in P \cup F_{\text{opt},p} \), the probability that \( p \in \mathcal{W}_i \) is at most \( 12\lambda \). Indeed, for any single \((1 \times 1)\) cell \( c \in G_0 \) in \([2\Delta]^2\), \( \mathbb{P}[\mathbb{P}[p \in c] \leq \frac{1}{2\Delta} \). Furthermore, the number of single cells \( c \) that are in \( \mathcal{W}_i \) is at most \( 12\lambda \cdot (2\Delta)^2 \). Hence, we have \( \mathbb{P}[p \in \mathcal{W}_i] \leq 12\lambda \).

For every \( p \in P \cup F_{\text{opt},p} \), let us define \( \text{val}(p) \) as follows:

\[
\text{val}(p) = \begin{cases} 
  f & \text{if } p \in F_{\text{opt},c} , \\
  d(p, F_{\text{opt},p}) & \text{if } p \notin F_{\text{opt},p} .
\end{cases}
\]

Observe that \( \text{OPT}_{\text{P}} = \sum_{p \in P \cup F_{\text{opt},p}} \text{val}(p) \). For every \( p \in P \cup F_{\text{opt},p} \), let \( X_p \) be the indicator random variable for the event \( p \in P \cup F_{\text{opt},p} \), and let \( Y_p \) be the indicator random variable for the event \( p \in \bigcup \mathcal{W}_i \). Notice that \( \mathbb{E}[X_p] = \mathbb{E}[Y_p] \).

Let us consider the contribution of a single point \( p \in P \cup F_{\text{opt},p} \) to \( \mathbb{E}\left[ \sum_{c \in \Lambda} \text{cost}_{\text{OPT},p}(B_c \cup Q_c) \right] \). If \( p \notin \bigcup_{c \in \Lambda} B_c \cup Q_c \), then the contribution is none; otherwise, the contribution is equal to \( \text{val}(p) \). Hence,

\[
\mathbb{E}\left[ \sum_{c \in \Lambda} \text{cost}_{\text{OPT},p}(B_c \cup Q_c) \right] \\
= \sum_{p \in P \cup F_{\text{opt},p}} \mathbb{E}[X_p] \cdot \text{val}(p) \\
\leq \sum_{p \in P \cup F_{\text{opt},p}} \mathbb{E}[Y_p] \cdot \text{val}(p) \\
\leq \sum_{p \in P \cup F_{\text{opt},p}} (12 \cdot \lambda \cdot (\log \Delta + 2)) \cdot \text{val}(p) \\
= 12 \lambda (\log \Delta + 2) \cdot \text{OPT}_{\text{P}} .
\]

2.3 Bounding the size of \( \Lambda \). Our next goal will be to estimate the two sums \( \sum_{c \in \Lambda} \text{cost}(I_c) \) and \( \sum_{c \in \Lambda} \text{cost}(B_c \cup Q_c) \) in terms of \( \text{OPT}_{\text{P}} \). For that we first bound (in Lemma 2.4 below) the number of cells in the partition \( \Lambda \), as a function of \( \text{OPT}_{\text{P}} \) and \( f \).

We begin with a simple claim that follows directly from the definition of light cells.

**Claim 2.3.** If the algorithm \( \mathcal{A} \) returns an \( \alpha \)-approximation of the cost of the facility location problem\(^3\), then for every

\[\text{cost}_{\mathcal{A}}(X) \leq \text{cost}_{\mathcal{A}}(X) \leq \alpha \cdot \text{cost}_{\mathcal{X}}(X) .\]

---

\(^3\)That is, for any set \( X \) of points in the plane, \( \mathcal{A} \) computes a value \( \text{cost}_{\mathcal{A}}(X) \) such that \( \text{OPT}_{\mathcal{X}} \leq \text{cost}_{\mathcal{A}}(X) \leq \alpha \cdot \text{OPT}_{\mathcal{X}} \).

---
light cell \( c \) and for every heavy cell \( c' \), we have \( \text{COST}(c) \leq \alpha N f \) and \( \text{COST}(c') > \gamma f \).

We call a cell \( c \in \bigcup_{i=0}^{\log \Delta + 1} G_i \) loaded if

\[
\text{COSTOPT}_x(c \cup Q_c) \geq (N - 8/\lambda) f.
\]

**Claim 2.4.** If \( c \) is a heavy cell, then \( c \) is loaded.

**Proof.** If \( c \) is a heavy cell, then by Lemma 2.2,

\[
\text{COST}(c) \leq \text{COSTOPT}_x(c \cup Q_c) + \frac{8f}{\lambda},
\]

and by Claim 2.3, we obtain \( \text{COST}(c) > \gamma f \). Hence,

\[
\text{COSTOPT}_x(c \cup Q_c) \geq \text{COST}(c) - \frac{8f}{\lambda} > \gamma f - \frac{8f}{\lambda}.
\]

We call a loaded cell \( c \) marked if either (i) \( c \in \Lambda \), or (ii) if \( c \) has no ancestor in \( \Lambda \) and no descendant of \( c \) is loaded. We will now state some basic properties of marked cells.

**Claim 2.5.** All marked cells are disjoint.

**Proof.** Let \( c \) and \( c' \) be any two marked cells, \( c \neq c' \). They are not disjoint only if one of them is an ancestor of another. If \( c, c' \in \Lambda \), then since \( \Lambda \) is a partition of \( [2\Delta]^2 \), we have \( c \) and \( c' \) disjoint. If none of \( c, c' \) is in \( \Lambda \), then since neither one of them can be an ancestor of another, again we have \( c \) and \( c' \) disjoint. Finally, if \( c \in \Lambda \) and \( c' \notin \Lambda \), then the definition of \( c' \) being marked implies that \( c \) cannot be an ancestor of \( c' \) because \( c \in \Lambda \), and \( c' \) cannot be a descendant of \( c' \) because \( c \) is loaded.

We have also the following claim.

**Claim 2.6.** If \( c \) is a loaded cell and has no proper ancestor in \( \Lambda \), then \( c \) has a marked descendant (which might be equal to \( c \)).

**Proof.** Let \( c \) be a loaded cell that has no proper ancestor in \( \Lambda \). Since \( \Lambda \) is a partition of the input space, it follows that \( c \) is either in \( \Lambda \) or it has a proper descendant in \( \Lambda \). In the former case, \( c \) is loaded and in \( \Lambda \) and so it is marked. In the latter case, let \( c' \) be the deepest descendant of \( c \) in \( \Lambda \) and let \( c' \) be the parent cell of \( c' \). Notice that by our choice of \( c' \) all subcells of \( c' \) must be in \( \Lambda \) (for otherwise \( c' \) would not be the deepest descendant of \( c \) in \( \Lambda \)). If one of these subcells is loaded it is also marked. Otherwise, \( c' \) has no loaded descendants and no ancestor in \( \Lambda \) and so it is marked. Hence, in all cases there is a marked descendant of \( c' \).

In our analysis, we will assign every cell \( c \in \Lambda \) to a marked cell \( \vartheta(c) \) as follows. If \( c \in \Lambda \) is marked, then \( \vartheta(c) = c \). Otherwise, let us consider the parent cell \( c^* \) of cell \( c \). Since cell \( c^* \) is heavy by the definition of \( \Lambda \), Claim 2.4 implies that \( c^* \) is loaded, and then Claim 2.6 ensures that \( c^* \) has a marked descendant. Let \( c' \) be any of the marked descendants of \( c^* \); we set \( \vartheta(c) = c' \).

**Claim 2.7.** For every cell \( c \in \Lambda \),

\[
\text{COSTOPT}_x(c \cup Q_c) \leq \text{COSTOPT}_x(\vartheta(c) \cup Q_{\vartheta(c)}) + (N - 8/\lambda) f.
\]

**Proof.** Observe that if \( c \in \Lambda \) is loaded, then our construction ensures that \( \vartheta(c) = c \), and hence

\[
\text{COSTOPT}_x(c \cup Q_c) = \text{COSTOPT}_x(\vartheta(c) \cup Q_{\vartheta(c)}).
\]

Otherwise, if \( c \in \Lambda \) is not loaded, then \( \text{COSTOPT}_x(c \cup Q_c) < (N - 8/\lambda) f \), and since \( \vartheta(c) \) is marked,

\[
\text{COSTOPT}_x(\vartheta(c) \cup Q_{\vartheta(c)}) \geq (N - 8/\lambda) f,
\]

from which the claim follows.

**Claim 2.8.** For every marked cell \( c \), the number of cells \( c' \in \Lambda \) for which \( \vartheta(c') = c \), is at most \( 3(\log \Delta + 1) + 1 \).

**Proof.** If \( c = \vartheta(c') \), then the parent of cell \( c' \) must be an ancestor of \( c \) (not necessarily a proper ancestor, that is, it is possible that \( c \) is the parent of \( c' \)). Hence, \( c' \) is a child of an ancestor of \( c \). Since there are at most \( \log \Delta + 2 \) ancestors of \( c \) and each of them (except the one at the most bottom level, in \( G_0 \)) has four children, there are at most \( 4(\log \Delta + 1) + 1 \) cells \( c' \) for which we may have \( \vartheta(c') = c \). (We have term “1” to count the case when \( c \in G_0 \).) We can reduce this bound further by observing that no ancestor \( c^* \) of \( c \) will have \( \vartheta(c^*) = c \), because all ancestors of \( c \) are loaded. Therefore, for every proper ancestor of \( c \) there are at most three children \( c' \) for which we may have \( \vartheta(c') = c \), and hence the number of possible cells \( c' \) for which we may have \( \vartheta(c') = c \) is upper bounded by \( 3(\log \Delta + 1) + 1 \).

**Claim 2.9.** For any set \( \mathcal{X} \) of disjoint cells in \( [2\Delta]^2 \),

\[
\sum_{c \in \mathcal{X}} \text{COSTOPT}_x(c \cup Q_c) \leq 5 \cdot \text{OPT}_x.
\]

**Proof.** This follows from the fact that every point \( p \in P \) appears in at most five terms in

\[
\sum_{c \in \mathcal{X}} \text{COSTOPT}_x(c \cup Q_c) + \sum_{c \in \mathcal{X}} \sum_{p \in P \cap (c \cup Q_c)} (d(p, F_{\text{OPT}_x}) + |F_{\text{OPT}_x} \cap (c \cup Q_c)|/f).
\]

Indeed, once if \( p \in c \), and at most four times if \( p \) is in the left, the right, the top, and the bottom boundary of some cell \( c' \).

With this, we are ready to prove the following upper bound on the number of cells in \( \Lambda \).

**Lemma 2.4.**

\[
|\Lambda| \leq \frac{15 \cdot \log \Delta + 2 \cdot \text{OPT}_x}{(N - \frac{8}{\lambda}) \cdot f}.
\]
Proof.

$$15 \cdot (\log \Delta + 2) \cdot \text{OPT}_P$$

$$\geq 3 \cdot (\log \Delta + 2) \cdot \sum_{\mathcal{c} \in \text{marked}} \text{COST}_{\text{OPT}_P}(\mathcal{c} \cup \mathcal{Q}_c)$$

$$\geq \sum_{\mathcal{c} \in \text{marked}} \text{COST}_{\text{OPT}_P}(\mathcal{c} \cup \mathcal{Q}_c) \cdot |\{\mathcal{c}' \in \Lambda : \vartheta(\mathcal{c}') = \mathcal{c}\}|$$

$$\geq \sum_{\mathcal{c} \in \text{marked}} \left((\mathcal{N} - \frac{6}{\lambda}) \cdot f\right) \cdot |\{\mathcal{c}' \in \Lambda : \vartheta(\mathcal{c}') = \mathcal{c}\}|$$

$$= (\mathcal{N} - \frac{6}{\lambda}) \cdot f \cdot |\Lambda|,$$

where the first inequality follows from Claims 2.5 and 2.9, the second one from Claim 2.8, the third one from the definition of loaded cells and the last identity follows from the fact that

$$\sum_{\mathcal{c} \in \text{marked}} |\{\mathcal{c}' \in \Lambda : \vartheta(\mathcal{c}') = \mathcal{c}\}| = |\Lambda|.$$

This yields the proof. \(\Box\)

### 2.4 Completing the proof of Partition Theorem 2.1.

Now we are ready to complete the proof of the Partition Theorem 2.1. We will first fix the constant \(\varrho\) for the probability bound, and then specify the values for \(\lambda\) and \(\mathcal{N}\): we have

$$\lambda = \frac{\varepsilon}{12 \cdot (1 + 3/\varrho) \cdot (\log \Delta + 2)}$$

and we will require that

$$\mathcal{N} \geq \frac{732 \cdot (1 + 3/\varrho)^2 \cdot (\log \Delta + 2)^2}{\varepsilon^2}.$$

Part (1) of the Partition Theorem 2.1 follows immediately from Claim 2.3 and from the fact that we have assumed that the algorithm \(\mathcal{A}\) used to define heavy and light cells is a constant-factor approximation algorithm for the cost facility location problem.

The proof of part (2) of the Partition Theorem 2.1 is split into two parts, one bounding \(\sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{I}_c)\) and another bounding \(\sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{B}_c)\).

In order to upper bound \(\sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{I}_c)\), we use Lemmas 2.1 and 2.4 to obtain the following:

$$\sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{I}_c) \leq \sum_{\mathcal{c} \in \Lambda} \left(\text{COST}_{\text{OPT}_P}(\mathcal{c}) + \frac{4f}{\lambda}\right)$$

$$= \text{OPT}_P + |\Lambda| \cdot \frac{4f}{\lambda}$$

$$\leq \text{OPT}_P + \frac{60 \cdot (\log \Delta + 2)}{\lambda \mathcal{N} - 8} \cdot \text{OPT}_P.$$

Thus, our bounds for \(\lambda\) and \(\mathcal{N}\) yield the following:

$$(2.2) \sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{I}_c) \leq (1 + \frac{\varepsilon}{1 + 3/\varrho}) \cdot \text{OPT}_P.$$

Next, we upper bound \(\sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{B}_c)\). We have the following:

$$E\left[\sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{B}_c)\right] \leq E\left[\sum_{\mathcal{c} \in \Lambda} \text{COST}_{\text{OPT}_P}(\mathcal{B}_c) + \frac{8f}{\lambda}\right]$$

$$\leq 12\lambda (\log \Delta + 2) \cdot \text{OPT}_P + \frac{8f|\Lambda|}{\lambda},$$

where the first inequality follows from Lemma 2.2 and the second one follows from Lemma 2.3. Then, we observe that since \(\lambda = \frac{\varepsilon}{12(1 + 3/\varrho)(\log \Delta + 2)}\), we obtain \(12\lambda (\log \Delta + 2) \cdot \text{OPT}_P \leq \frac{\varepsilon}{1 + 3/\varrho} \cdot \text{OPT}_P\).

Further, \(|\Lambda| \cdot \frac{f}{\varrho} \leq \frac{\varepsilon}{1 + 3/\varrho} \cdot \text{OPT}_P\) from our analysis above (inequality (2.2)), and hence \(\frac{8f|\Lambda|}{\lambda} \leq \frac{2\varepsilon}{1 + 3/\varrho} \cdot \text{OPT}_P\). This yields

$$E\left[\sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{B}_c)\right] \leq \frac{3}{1 + 3/\varrho} \cdot \varepsilon \cdot \text{OPT}_P.$$

Therefore, Markov inequality implies that with probability at least \(1 - \varrho\) holds

$$(2.3) \sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{B}_c) \leq \frac{3}{\varrho + 3} \cdot \varepsilon \cdot \text{OPT}_P.$$

Now the proof of part (2) of the Partition Theorem 2.1 follows directly from inequalities (2.2) and (2.3), and from the fact that for every cell \(\mathcal{c}\) we have

$$\text{COST}(\mathcal{c}) \leq \text{COST}(\mathcal{I}_c) + \text{COST}(\mathcal{B}_c).$$

If we sum up these bounds, we obtain that with probability at least \(1 - \varrho\) the following holds

$$\sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{c}) \leq \sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{I}_c) + \sum_{\mathcal{c} \in \Lambda} \text{COST}(\mathcal{B}_c)$$

$$\leq (1 + \frac{\varepsilon}{1 + 3/\varrho}) \cdot \text{OPT}_P + \frac{3}{\varrho + 3} \varepsilon \cdot \text{OPT}_P$$

$$= (1 + \varepsilon) \cdot \text{OPT}_P. \Box$$

### 3 Fast \((1 + \varepsilon)\)-approximation for facility location

With Theorem 2.1 at hand, we design our fast \((1 + \varepsilon)\)-approximation for the facility location problem and show that it runs in \(O(n \log n \log \Delta \log \log n + (\log \log(n \Delta))^8)\) time (assuming that \(\varepsilon\) is an arbitrary constant).

In this section, to define heavy and light cells in the partitioning scheme, we will use as \(\mathcal{A}\) a deterministic \(\alpha\)-factor approximation algorithm for the facility location problem, as developed later in Section 4, in Theorem 4.1. This algorithm runs in time \(O(n \log n \log \log n)\) and ensures that \(\alpha = O(1)\).
Find a partition $\Lambda$ of $[2\Delta]^2$ as promised in Theorem 2.1, using as $A$ the algorithm from Section 4.

For each cell $c \in \Lambda$: Find a $(1 + \varepsilon)$-approximation for the facility location problem for $c$.

Return the union of the obtained solutions.

From now on, let $|P| = n$. We have three simple claims that describe properties of the algorithm.

**Claim 3.1.** Partition $\Lambda$ of $[2\Delta]^2$ can be found in $O(n \log \Delta \log n \log \log n)$ time.

**Proof.** To find the partition $\Lambda$, we have to go through the nested grid $G_0, G_1, \ldots, G_{\log \Delta + 1}$ and check if the appropriate cells are heavy or not. This requires calls to the algorithm $A$, where in each level, we have to consider some set of cells with the total number of points being at most $n$. Hence, the total running time is $(\log \Delta + 2) \times O(n \log n \log \log n)$, by Theorem 4.1. 

Our next claim follows directly from an earlier PTAS for facility location due to Kolliopoulos and Rao [17]. (Let us recall that for any cell $c$, the number of points from $P$ inside $c$ is denoted by $n_c$.)

**Claim 3.2.** For any cell $c \in \Lambda$ one can find a $(1 + \varepsilon)$-approximation for facility location for $c$ in time $O(n_c \log^8 n_c \log n 2^{O((\log(1/\varepsilon))/\varepsilon))}$.

the algorithm is randomized and it errs with probability at most $1/n^2$.

**Proof.** The claims follows directly from the result due to Kolliopoulos and Rao [17] for the facility location problem that gives a randomized algorithm that “with high probability” finds a $(1 + \varepsilon)$-approximation for the facility location problem for $c$ in time $O(n_c \cdot \log^8 n_c \cdot 2^{O((\log(1/\varepsilon))/\varepsilon))}$. However, since the error probability is proportional to the size of the input $n_c$, we have to repeat this algorithm $O(\log n)$ times and then choose the best solution, to ensure that the error probability is at most $1/n^2$.

Next, we note that our construction of $\Omega$ implies that $k^+ \leq (1 + \varepsilon)k_r$. Therefore, if we combine these two claims, then we obtain:

$$
\sum_{p \in P \cap c} d(p, Q_{k^+}) + k^+ \cdot f \\ 
\leq (1 + \varepsilon) \sum_{p \in P \cap c} d(p, F_{opt, p}(\varepsilon)) + (1 + \varepsilon) \cdot k_c \cdot f \\
= (1 + \varepsilon) \cdot \text{COST}(\varepsilon)
$$
Since by our choice of \( k^* \):
\[
\sum_{p \in P \cap c} d(p, Q_{k^*}) + k^* \cdot f \leq \sum_{p \in P \cap c} d(p, Q_k) + k^* \cdot f ,
\]
this implies that \( Q_{k^*} \) is a \((1 + \varepsilon)\)-approximation for the facility location problem for \( c \).

Now, we analyze the running time of our algorithm. We run \(|\mathcal{D}|\) times a \((1 + \varepsilon)\)-approximation algorithm for the \( k\)-median problem for an instance with \( n_\varepsilon \) points, \( n_\varepsilon = |P \cap c| \). Har-Peled and Mazumdar \cite{Har-PeledM11} show how this problem can be solved in time \( O(n_\varepsilon + 2^{O((1+\log(1/\varepsilon))/\varepsilon)} \cdot \log \log n_\varepsilon) \). However, to achieve the required error probability at most \( 1/n^2 \), we will repeat the algorithm by Har-Peled and Mazumdar \cite{Har-PeledM11} \( \mathcal{O}(\log n) \) number of times, and we will choose the best solution. Since in our case \( k \leq r \), the total running time of the algorithm is bounded by
\[
\mathcal{O}((n_\varepsilon + 2^{O((1+\log(1/\varepsilon))/\varepsilon)} \cdot (r \log n)^{O(1)} \cdot \log n) \times |\mathcal{D}| ,
\]
as required. \( \square \)

With Claims 3.1–3.3 at hand, we are ready to prove our main theorem.

**Theorem 3.1.** Algorithm PTAS\((P)\) with probability at least \( \frac{2}{3} \) finds a \((1 + O(\varepsilon))\)-approximation for the facility location of \( P \) in time \( \mathcal{O}(n \cdot \log n \cdot \log \Delta \cdot \log \log n + n \cdot \log n \cdot (\log \log (n\Delta))^8 \cdot 2^{O((\log(1/\varepsilon))/\varepsilon)}) \).

**Proof.** Theorem 2.1 and Claim 3.1 ensure that in time \( \mathcal{O}(n \log \Delta \log \log n) \) one can find a partition \( \Lambda \) such that with probability at least \( \frac{2}{3} \), \( \Lambda \) satisfies properties (1) and (2) of Theorem 2.1.

Next, we consider the time needed to find feasible solutions for the facility location problem for all cells \( c \in \Lambda \), whose cost is (with probability at least \( 1 - \frac{1}{n} \)) upper bounded by \((1 + \varepsilon) \cdot \text{COST}(c)\) for all \( c \in \Lambda \). We will use two alternative ways of doing this, using either Claim 3.2 or 3.3, depending on the size of \( P \cap c \).

Let \( c \) be a positive constant such that the running time of the algorithm in Claim 3.3 is
\[
\mathcal{O}(n_\varepsilon \cdot \log n \cdot \log_1 r + 2^{O((1+\log(1/\varepsilon))/\varepsilon)} \cdot (r \log n)^c .
\]

Let \( \zeta = 2^{O((1+\log(1/\varepsilon))/\varepsilon)} \cdot (r \log n)^c \). We define \( \Lambda_S \) to be the subset of cells \( c \in \Lambda \) with \( n_\varepsilon \leq \zeta \) and \( \Lambda_L \) as the subset of cells \( c \in \Lambda \) with \( n_\varepsilon > \zeta \) (notice that \( \Lambda_L = \Lambda \setminus \Lambda_S \)).

For each cell \( c \in \Lambda_S \), we will find a \((1 + \varepsilon)\)-approximate solutions for the facility location problem using Claim 3.2, and for the cells \( c \in \Lambda_L \), we will find a \((1 + \varepsilon)\)-approximate solutions for the facility location problem using Claim 3.3. Since we perform at most \( n \) calls to the randomized algorithms in Claims 3.2 and 3.3, the probability that we will find \((1 + \varepsilon)\)-approximate solutions for the facility location problem for all cells in \( \Lambda \) is at least \( 1 - 1/n \). Therefore, by Theorem 2.1, the total cost of the union of the solutions to these facility location problems is (with probability at least \( \frac{3}{4} - \frac{1}{n} \)) bounded by:
\[
\sum_{c \in \Lambda} (1 + \varepsilon) \cdot \text{COST}(c) \leq (1 + \varepsilon)^2 \cdot \text{OPT} \\
\leq (1 + O(\varepsilon)) \cdot \text{OPT} .
\]

Now, we must analyze the running time needed to approximate the cost of facility location for all cells in \( \Lambda \). The running time of approximating the cost of facility location for the cells in \( \Lambda_S \) is by Claim 3.2 upper bounded by the following:
\[
\sum_{c \in \Lambda_S} \mathcal{O}(n_\varepsilon \cdot \log n_\varepsilon \cdot \log n) \cdot 2^{O((\log(1/\varepsilon))/\varepsilon)} \\
\leq \sum_{c \in \Lambda_S} \mathcal{O}(n_\varepsilon \cdot \log^8 \zeta \cdot \log n) \cdot 2^{O((\log(1/\varepsilon))/\varepsilon)} \\
\leq \mathcal{O}(n \cdot \log^8 \zeta \cdot \log n) \cdot 2^{O((\log(1/\varepsilon))/\varepsilon)} .
\]

The running time of approximating the cost of facility location for the cells in \( \Lambda_L \) is by Claim 3.3 upper bounded by
\[
\sum_{c \in \Lambda_L} \mathcal{O}(n_\varepsilon \cdot \log n \cdot \log_1 r + \zeta) ,
\]
with \( r = \mathcal{O}((\log \Delta)/\varepsilon)^2 \).

Then, the definition of \( \Lambda_L \) implies that \( |\Lambda_L| \leq n/\zeta \) and hence, the running time is upper bounded by the following:
\[
\sum_{c \in \Lambda_L} \mathcal{O}(n_\varepsilon \cdot \log n \cdot \log_1 r + \zeta) \\
= \mathcal{O}(n \cdot \log n \cdot \log_1 r + \zeta) = \mathcal{O}(n \cdot \log n \cdot \log_1 (\log \Delta/\varepsilon) + \mathcal{O}(|\Lambda_L| \cdot \zeta) \\
(3.5) = \mathcal{O}(n \cdot \log n \cdot \log_1 (\log \Delta/\varepsilon) + \mathcal{O}(|\Lambda_L| \cdot \zeta)) .
\]

Therefore, inequalities (3.4) and (3.5) imply that the running time of approximating the cost of facility location for all cells in \( \Lambda \) using Claims 3.2 and 3.3 is upper bounded by \( \mathcal{O}(n \cdot \log^8 \zeta \cdot \log n) \cdot 2^{O((\log(1/\varepsilon))/\varepsilon)} + \mathcal{O}(n \cdot \log n \cdot \log_1 (\log \Delta/\varepsilon)) \), which is asymptotically equal to
\[
\mathcal{O}(n \cdot \log n \cdot (\log \log (n\Delta))^8) \cdot 2^{O((\log(1/\varepsilon))/\varepsilon)} ,
\]
what completes the proof. \( \square \)

**Remark 3.1.** Theorem 3.1 has been presented in the framework convenient for our streaming algorithms and for some natural applications. However, one can extend Theorem 3.1 to the problem of facility location on the plane, where the input point set \( P \) and the set of facilities are allowed to be at any location in \( \mathbb{R}^2 \) (not just in the grid \( \lfloor \Delta \rfloor^2 \)); see, e.g., \cite[Section 3.2.3]{Li04} and \cite[Section 2]{Pan05}, for more details of the reduction.
THEOREM 3.2. For any \( \varepsilon > 0 \), there is an algorithm that for any set \( P \) of \( n \) points in \( \mathbb{R}^2 \) with probability at least \( \frac{2}{3} \), finds a \((1 + \varepsilon)\)-approximation for the facility location of \( P \) in time

\[
O(n \log n \log \log n + (\log \log n)^{8\varepsilon} 2^{O((1/\varepsilon)/\varepsilon))}.
\]

4 Fast constant-factor approximation algorithm for facility location

Our PTAS in Section 3 requires a fast constant-factor approximation algorithm for the uniform facility location problem. In this section we present such a fast \((O(n \log n \log \log n))\)-time constant-factor approximation algorithm. The algorithm is a simple modification of an algorithm by Mettu and Plaxton [21] and its extensions in [4, 6].

Let \( P \subseteq \mathbb{R}^2 \) be a set of \( n \) points in the plane. In this section, we will assume, without loss of generality, that the input is scaled such that \( f = 1 \). Hence, we have

\[
\text{COST}(P, F) = |F| + \sum_{p \in P} \min_{q \in F} d(p, q).
\]

The algorithm requires the following definitions of the \((\ell_\infty)\)-radius of a point. Let \( Q(q, m) \) be the square centered at \( q \) with side length \( 2m \) (i.e., \( Q(q, m) \) is the \( \ell_\infty \)-ball with radius \( m \)). Given \( q \in P \), the radius of \( q \), denoted by \( R(q) \), is the smallest value \( m \) such that

\[
\sum_{p \in P \cap Q(q, m)} \min_{i=1,2} |p^{(i)} - q^{(i)}| + m = f,
\]

where \( p^{(i)} \) and \( q^{(i)} \) refer to the \( i \)-th coordinate of \( p \) and \( q \), respectively. In other words, the radius of \( q \) is the value \( R(q) \) such that the sum of distances of the points inside the square \( Q(q, m) \) to the boundary of \( Q(q, m) \) equals the opening cost \( f \).

The discrete radius of \( q \in P \), denoted by \( r(q) \), is the smallest value \( m = 2^i \), \( i \in \mathbb{Z} \), such that there are at least \( 1/m \) points inside \( Q(q, m) \). The definition of radius is an adaptation of a similar definition in [21]. It had been extended to \( \ell_\infty \)-balls in [6]. The definition of discrete radius is similar to a definition in [4].

We first prove that the discrete radius approximates the radius of a point up to a constant factor. (A similar lemma for balls in metric spaces is in [4]. Our lemma is a simple adaption to our setting.)

LEMMA 4.1. Let \( q \in P \) be an arbitrary point. Then, we have

\[
\frac{1}{2} \cdot R(q) \leq r(q) < 2 \cdot R(q).
\]

Proof. The first inequality follows from the fact that, inside a square of radius \( 2r(q) \), we have at least \( 1/r(q) \) points whose distance to the border is at least \( r(q) \). Hence, \( R(q) \leq 2r(q) \).

By the definition of the discrete radius, we know that there are less than \( 2/r(q) \) points inside the square with center \( q \) and radius \( r(q)/2 \). Each of the points has a distance of at most \( r(q)/2 \) to the boundary of the square. Hence, \( R(q) > r(q)/2 \), which proves the second inequality.

Thus, the discrete radius is a \( 4 \)-approximation of the radius of a point.

Next, we can state the algorithm.

<table>
<thead>
<tr>
<th>FACILITYLOCATION(P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• For each point ( q \in P ) do</td>
</tr>
<tr>
<td>• Compute the discrete radius ( r(q) ) of ( q )</td>
</tr>
<tr>
<td>• Sort the discrete radii in non-decreasing order</td>
</tr>
<tr>
<td>• For each point ( q ) in this order do</td>
</tr>
<tr>
<td>• If there is no open facility within distance at most ( 8r(q) ) then ( F = F \cup q )</td>
</tr>
<tr>
<td>• Return ( F )</td>
</tr>
</tbody>
</table>

In the rest of this section, we first show that Algorithm FACILITYLOCATION computes a \( O(1) \)-approximation and then we address how it can be implemented in \( O(n \log n \log \log n) \) time.

We observe that, for any \( q_1, q_2 \in F \), we have that the squares \( Q(q_1, 4r(q_1)) \) and \( Q(q_2, 4r(q_2)) \) are disjoint. This together with Lemma 4.1 implies that the squares \( Q(q_1, R(q_1)) \) and \( Q(q_2, R(q_2)) \) are disjoint for any \( q_1, q_2 \in F \).

We will show that, for any solution \( F \) computed by our algorithm,

\[
\text{COST}(P, F) = |F| + \sum_{p \in P} \min_{q \in F} d(p, q) \leq 17 \sum_{p \in P} R(p).
\]

Then, we will prove that the cost of an optimal solution is more than \( \sum_{p \in P} R(p)/16 \). This proves that the algorithm is a constant-approximation algorithm. Both proofs are similar to earlier proofs in [4] and based on the work in [21].

LEMMA 4.2. Let \( F \) be a set of facilities computed by Algorithm FACILITYLOCATION on input \( P \). Then, we have

\[
\text{COST}(P, F) \leq 17 \sum_{p \in P} R(p).
\]

Proof. A simple consequence of our algorithm is that any point \( p \in P \) has a distance of at most \( 8r(p) \) to the nearest open facility. Furthermore, by Lemma 4.1, we know that

\[
r(p) \leq 2R(p).
\]

Hence, any point \( p \in P \) has connection cost at most \( 16R(p) \). To proceed, we need the following claim.
CLAIM 4.1. Let $F$ be a set of facilities computed by algorithm FACILITYLOCATION. Then, for any $q \in F$, we have

$$\sum_{p \in P \cap Q(q,R(q))} R(p) \geq 1.$$ 

Proof. Let $p \in P$ be an arbitrary point inside $Q(q,R(q))$, and let $\ell$ be its distance to the boundary of $Q(q,R(q))$. Then, the square $Q(p,\ell)$ is completely contained in $Q(q,R(q))$. This implies that, for any point $r \in P \cap Q(p,\ell)$, the distance to the boundary of $Q(p,\ell)$ is at most as large as the distance to the boundary of $Q(q,R(q))$. This implies that $R(p) \geq \ell$. This together with the definition of $R(q)$ implies the claim. \qed

We have already observed that, for any $q_1,q_2 \in F$, the squares $Q(q_1,R(q_1))$ and $Q(q_2,R(q_2))$ are disjoint. Now, the above claim implies that $|P| \leq \sum_{p \in P} R(p)$. Hence, the lemma follows. \qed

Our next step is to prove that the cost of an optimal solution $F_{opt}$ is $\Omega(\sum_{p \in P} R(p))$.

LEMMA 4.3. Let $F_{opt}$ be an optimal solution. Then, we have

$$\text{COST}(P,F_{opt}) > \sum_{p \in P} R(p)/16.$$ 

Proof. We divide $P$ into two subsets $P_A$ and $P_B$. The subset $P_A$ contains all points $p$ whose distance to $F_{opt}$, is at least $r(p)/8$. The subset $P_B$ contains the remaining points. For a facility $q \in F_{opt}$, let $P_B(q)$ be the subset of points in $P_B$ for which $q$ is the closest facility in $F_{opt}$, breaking ties arbitrarily. The subset $P_B(q)$ contains point that are rather close to $q$ with respect to their radius. This implies that for two point $p,t \in P_B(q)$ the squares must have a big intersection, which will be exploited for the following claim.

CLAIM 4.2. Let $q \in F_{opt}$ be an arbitrary facility, and let $p,t \in P_B(q)$. Then we have

$$d(p,t) < r(p)/2.$$ 

Proof. Let $p,t$ be arbitrary points from $P_B(q)$. By the triangle inequality, we have $d(p,t) \leq d(p,q) + d(q,t)$. From the definition of $P_B(q)$ and $p \in P_B(q)$, we know that $d(q,p) < r(p)/8$. Thus, the claim follows if we can prove that $d(q,t) \leq 3r(p)/8$.

Assume, for the purpose of contradiction, that $d(q,t) > 3r(p)/8$. Since $t \in P_B(q)$ we know that $d(q,t) < r(t)/8$. This, together with the previous inequality, implies $r(t) > 3r(p)$. Since $r(t)$ is a power of 2, we know that $r(t) \geq 4r(p)$ and so $r(t)/4 \geq r(p)$. Due to triangle inequality and the above arguments, we obtain

$$d(t,p) + r(p) \leq d(t,q) + d(q,p) + r(p) < r(t)/2.$$ 

Hence, the square $Q(t,r(t)/2)$ contains the square $Q(p,r(p))$. This is a contradiction, because then the radius of $t$ would be at most $r(t)/2$. \qed

Let $t$ be the point in $P_B(q)$ with biggest discrete radius, i.e., $r(p) \leq r(t)$ for any $p \in P_B(q)$. By the definition of the discrete radius, we know that

$$|Q(t,r(t)/2)| < 2/r(t).$$

Furthermore, due to Claim 4.2, all the points in $P_B(q)$ are contained in $Q(t,r(t)/2)$. This implies that

$$2 > |Q(t,r(t)/2)| \cdot r(t) \geq \sum_{p \in P_B(q)} r(p) \geq \sum_{p \in P_B(q)} R(p)/2,$$

where the last inequality follows from Lemma 4.1. Hence, we get

$$|F_{opt}| > \sum_{q \in F_{opt}} \sum_{p \in P_B(q)} R(p)/4.$$ 

Since

$$\sum_{p \in P_A} \min_{q \in F_{opt}} d(p,q) \geq \sum_{p \in P_A} R(p)/16,$$

we obtain the following inequality,

$$\text{COST}(P,F_{opt}) = |F_{opt}| + \sum_{p \in P} \min_{q \in F_{opt}} d(p,q)$$

$$\geq \sum_{p \in P_B} R(p)/4 + \sum_{p \in P_A} R(p)/16$$

$$\geq \sum_{p \in P} R(p)/16,$$

which completes the proof of Lemma 4.3. \qed

We conclude by Lemmas 4.1, 4.2, and 4.3 that Algorithm FACILITYLOCATION yields a constant approximation.

It remains to discuss how to implement the algorithm. For this purpose, we build a layered range tree the input point set $P$. Such a range tree can be build in $O(n \log n)$ time and can be used to answer orthogonal range counting queries in $O(\log n)$ time.

Now, observe that, for any $p \in P$, we have

$$1/2^{\lfloor \log n \rfloor} \leq r(p) \leq 1.$$
Thus, there are only \(O(\log n)\) possible values for \(r(p)\). We can use the range tree to compute the number of points inside a square of radius \(1/2^i\) centered at \(p\) and use this to determine the radius of point \(p\). We further observe that we can use binary search to speed up this process using only \(O(\log \log n)\) queries to the range tree.

Hence, computing the discrete radii of all the points in \(P\) requires \(O(n \log n \log \log n)\) time. Afterwards, we sort the radii in non-decreasing order in \(O(n \log n)\) time.

Next, we partition \(P\) into subsets \(P_0, \ldots, P_{|P|}\), where \(P_i\) is the set of points whose radius is \(1/2^i\). For each \(P_i\), we construct a dynamic 2D range tree, which supports queries in \(O(\log n + k)\) time, where \(k\) is the number of points, and insertions and deletions in \(O(\log n)\) time.

Then, we start with a point \(q \in P\) with smallest radius. We insert \(q\) into our set of facilities \(F\) and use the range trees to identify all points \(p\) with \(r(p) \geq r(q)\) that contain \(q\) inside \(Q(p, 8r(p))\). This can be done as follows. For each \(i\) with \(1/2^i \geq r(q)\), we perform a range reporting query in the range tree for \(P_i\) for the square with radius \(8/2^i\) centered at \(q\) in the \(i\)-th range tree. By symmetry, this contains all points \(p\) with \(r(p) = 1/2^i \geq r(q)\) that contain \(q\) inside \(Q(p, 8r(p))\). We delete these points from their corresponding range tree. Since each point is deleted at most once from a range tree, the total time required for the deletions is \(O(n \log n)\). It remains to analyze the time for the range reporting queries. Each range reporting query takes \(O(\log n + k)\) time, where \(k\) is the number of reported points. However, the \(k\) can be charged to the deletions since every reported point will be deleted. Thus, in the following we may assume that each query takes \(O(\log n)\) time.

Note that, for each facility \(q\) in \(P_i\) that we put into \(F\), at least all the points in \(Q(q, r(q))\) will be deleted. Since, for any \(q' \in F\setminus\{q\}\) the squares \(Q(q, r(q))\) and \(Q(q', r(q'))\) do not overlap and the number of points in \(Q(q, r(q))\) is at least \(2^i\), there are at most \(n/2^i\) facilities in \(F \cap P_i\).

Thus, the overall running time is

\[
\mathcal{O}(n \log n) + \sum_{i=0}^{|P|} \frac{n}{2^i} (i + 1) \cdot \mathcal{O}(\log n) = \mathcal{O}(n \log n).
\]

Hence, the total running time of the algorithm is \(O(n \log n \log \log n)\).

Therefore, we can conclude the discussion in this section with the following theorem.

**Theorem 4.1.** Given a set \(P \subseteq \mathbb{R}^2\) of \(n\) points in plane, Algorithm FacilityLocation computes a constant-factor approximation of the uniform facility location problem for \(P\) in \(O(n \log n \log \log n)\) time.

### 5 A streaming algorithm

In this section, we derive a \((1 + \varepsilon)\)-approximation algorithm in the dynamic geometric streaming setting, where the items of the stream are insertions and deletions of points from \(\Delta^2 = \{1, \ldots, \Delta\}^2\), and where \(1 > \varepsilon > 0\) is an arbitrary fixed real. We will assume \(\Delta \geq 4\).

Let us fix a constant failure probability bound \(\rho\), \(0 < \rho < 1\). Let \(\kappa^*\) be the minimal value for \(\kappa\) for which Partition Theorem 2.1 holds with the failure probability \(\rho/3\); note that

\[
\kappa^* = \Theta \left(\frac{\log \Delta + 2}{\varepsilon}\right) ^{2}.
\]

Let \(\kappa = 2 \cdot \kappa^* \cdot f\) (and thus, Partition Theorem 2.1 will hold with the probability at least \(1 - \rho/3\) for any choice of the threshold defining heavy and light cells that is greater than or equal to \(\kappa/2\), assuming the algorithm \(\mathcal{A}\) used is an exact algorithm (and hence with \(\alpha = 1\)).

Our streaming algorithm will use the partitioning scheme as presented in Section 2, and in particular, Partition Theorem 2.1. For reasons that will become clear later, to distinguish between light and heavy cells we define a random threshold \(T\) as \(T = \kappa \cdot \tau_{\text{rand}}\), where \(\tau_{\text{rand}}\) is chosen uniformly at random from the interval \([1, 2]\). We use the threshold \(T\) to distinguish between heavy and light cells in the randomly shifted nested grid partition \(G_0, G_1, \ldots, G_{\log \Delta + 1}\) (cf. Section 3). Any cell \(c\) with \(\text{COST}(c) > T\) will be called heavy and any cell with \(\text{COST}(c) \leq T\) will be called light. A light cell whose parent cell is heavy will be called a maximal light cell. (In our analysis, we will frequently use the fact that independently of the random choice of \(\tau_{\text{rand}}\), if \(\text{COST}(c) \leq \kappa/2\) then \(c\) is light, and if \(\text{COST}(c) > 2\kappa\) then \(c\) is heavy.)

5.1 **Focusing on \(\delta\)-detectable cells.** For a parameter \(\delta\), a maximal light cell \(c\) with parent cell \(c^*\) is called \(\delta\)-detectable if

\[
\text{COST}(c) \leq (1 - \delta) T + (1 + \delta) T \leq \text{COST}(c^*)\,.
\]

Our first lemma proves that for sufficiently small \(\delta\), with constant probability the vast majority of the maximal light cells are \(\delta\)-detectable and as the result, we can “ignore” the cost of maximal light cells that are not \(\delta\)-detectable.

**Lemma 5.1.** Let \(T\) be chosen as stated above and let \(\rho\) be an arbitrary positive constant. If \(\delta = c \cdot \varepsilon / \log \Delta\), for an appropriate constant \(c = c(\rho)\), then with probability at least \(1 - \rho\) the following inequalities hold:

\[
\sum_{i=0}^{\log \Delta + 1} \sum_{c \in G_i} \text{COST}(c) \leq (1 + \varepsilon) \cdot \text{OPT}.
\]
and
\[(1 - \varepsilon) \cdot \text{OPT} \leq \sum_{i=0}^{\log \Delta + 1} \sum_{\delta\text{-detectable cell } c \in G_i} \text{COST}(c) .\]

**Proof.** The first bound (the upper bound) follows from the Partition Theorem 2.1 (where will set the failure probability to be equal to \(g/3\) that ensures that with probability at least \(1 - g/3\) we have
\[\sum_{\text{maximal light cells } c \in \bigcup_{i=0}^{\log \Delta + 1} G_i} \text{COST}(c) \leq (1 + \varepsilon) \cdot \text{OPT} .\]

Hence, with probability at least \(1 - g/3\) the following holds:
\[\sum_{i=0}^{\log \Delta + 1} \sum_{\delta\text{-detectable cell } c \in G_i} \text{COST}(c) \leq \sum_{\text{maximal light cells } c \in \bigcup_{i=0}^{\log \Delta + 1} G_i} \text{COST}(c) \leq (1 + \varepsilon) \cdot \text{OPT} .\]

Now we proceed to the second inequality (the lower bound). We begin with basic definitions.

- We define \(Q\) to be the set of cells \(c \in G_0, \ldots, G_{\log \Delta + 1}\) that would be maximal light if the threshold was chosen as \(\kappa/2\). That is, a cell \(c\) is in \(Q\) if and only if
  \[\text{COST}(c) \leq \kappa/2\text{ and } \text{COST}(c^*) > \kappa/2 ,\]
  where \(c^*\) is the parent cell of \(c\).

- Let \(MQ\) be the set of all cells \(c \in G_0, \ldots, G_{\log \Delta + 1}\) with \(\text{COST}(c) \leq 2\kappa\) such that either \(c \in Q\) or \(c\) is an ancestor of a cell in \(Q\).

We observe that only the cells in \(MQ\) can be maximal light (when the threshold \(T = \kappa \cdot \text{rand}\) is chosen, as above). Indeed, firstly, any cell \(c\) that is maximal light with the threshold \(T\) must be light, and hence satisfy \(\text{COST}(c) \leq T \leq 2\kappa\), and secondly, any such cell \(c\) is either in \(Q\), or is an ancestor of a cell in \(Q\).

Next, we have the following claim.

**CLAIM 5.1.** *With probability at least 1 - \(g/3\) we have that\*[\(c \in MQ\)]
\[\sum_{c \in MQ} \text{COST}(c) = O(\log \Delta) \cdot \text{OPT} .\]

**Proof.** Let \(BQ\) be the set of cells from \(MQ \setminus Q\) that do not have a descendant in \(MQ \setminus Q\). Observe that every cell \(c\) in \(BQ\) has \(\kappa/2 < \text{COST}(c) \leq 2\kappa\), and since no descendant of \(c\) has cost greater than \(\kappa/2\), we conclude that all four children of \(c\) are in \(Q\).

Therefore, since the cost of any cell is upper bounded by the sum of the costs of its children, we have
\[\sum_{c \in BQ} \text{COST}(c) \leq \sum_{c \in Q} \text{COST}(c) .\]

Next, we observe that every cell \(c\) in \(MQ \setminus Q\) is either in \(BQ\) or has a descendant in \(BQ\). Also, every cell has at most \(\log \Delta + 1\) ancestors and every cell \(c \in MQ \setminus Q\) has \(\kappa/2 < \text{COST}(c) \leq 2\kappa\). Thus, we can conclude that
\[\sum_{c \in MQ \setminus Q} \text{COST}(c) \leq \sum_{c \in BQ} (\log \Delta + 1) \cdot (4 \cdot \text{COST}(c)) \leq 4 \cdot (\log \Delta + 1) \cdot \sum_{c \in Q} \text{COST}(c) .\]

Hence,
\[\sum_{c \in MQ} \text{COST}(c) = \sum_{c \in MQ \setminus Q} \text{COST}(c) + \sum_{c \in Q} \text{COST}(c) \leq (4 \cdot (\log \Delta + 1) + 4) \cdot \sum_{c \in Q} \text{COST}(c) .\]

Now, to conclude the proof of Claim 5.1, we observe that by taking maximal light cells for the threshold \(T = \kappa/2\), Theorem 2.1 ensures that with probability at least \(1 - g/3\) we get
\[\sum_{c \in Q} \text{COST}(c) \leq (1 + \varepsilon) \cdot \text{OPT} .\]

We will use Claim 5.1 to conclude the proof of the lower bound in Lemma 5.1. Let us first assume that \(\sum_{c \in MQ} \text{COST}(c) = O(\log \Delta \cdot \text{OPT})\), what by Claim 5.1 holds with probability at least \(1 - g/3\). As observed above, every maximal light cell \(c\) that is not \(\delta\)-detectable must be in \(MQ\). Notice that the definition of the set \(MQ\) has been done independently of the choice of \(\text{rand}\) in the definition of the threshold \(T\), but the choice of \(\text{rand}\) is essential for being \(\delta\)-detectable.

By definition, if a cell \(c\) with parent cell \(c^*\) is not \(\delta\)-detectable, then we must have \((1 - \delta)T < \text{COST}(c)\) or \(\text{COST}(c^*) < (1 + \delta)T\), and hence
\[\min\{|T - \text{COST}(c)|, |T - \text{COST}(c^*)|\} \leq \delta T .\]

Next, we characterize the values \(|T - \text{COST}(c)|\) and \(|T - \text{COST}(c^*)|\) with respect to the random choice of \(\text{rand}\) that defines \(T = \kappa \cdot \text{rand}\). Since we choose \(\text{rand}\) uniformly at random from \([1, 2]\), the probability that \(|T - \text{COST}(c)| \leq \delta T\) is at most \(2\delta\), and so is the probability that \(|T - \text{COST}(c^*)| \leq \delta T\). Thus, in expectation at most an \((\delta/2)\)-fraction of cells in \(MQ\) are not \(\delta\)-detectable. Hence, by Markov’s inequality and by our assumption that
\[\sum_{c \in MQ} \text{COST}(c) = O(\log \Delta \cdot \text{OPT}) ,\]
there is a constant $c > 0$ such that for every $t > 0$,
\[
\Pr \left[ \sum_{\text{not } \delta\text{-detectable cell } c \in MQ} \text{COST}(c) \geq t \right] \leq \frac{4\delta \sum_{c \in MQ} \text{COST}(c)}{t} \leq \frac{c \cdot \delta \cdot \log \Delta \cdot \text{OPT}}{t} .
\]

Hence, for a given $\varrho > 0$, if we set $\delta = \frac{\varrho}{\log \Delta + 1}$, then (by taking $t = \varrho \cdot \text{OPT}$ in the bound above) we obtain that with probability at least $1 - \varrho/3$ it holds:
\[
\sum_{\text{not } \delta\text{-detectable cell } c \in MQ} \text{COST}(c) \leq \varrho \cdot \text{OPT} .
\]

This implies that conditioned on the bound in Claim 5.1 that $\sum_{c \in MQ} \text{COST}(c) = O(\log \Delta \cdot \text{OPT})$, we have with probability at least $1 - \varrho/3$ the following bound:
\[
\sum_{c \text{ is } \delta\text{-detectable cell } c \in \bigcup_{i=0}^{\log \Delta + 1} G_i} \text{COST}(c) \geq \text{OPT} - \sum_{\text{not } \delta\text{-detectable cell } c \in MQ} \text{COST}(c) \geq (1 - \varrho/3) \cdot \text{OPT} ,
\]

where the first inequality follows from the fact that
\[
\sum_{c \text{ is maximal light cell } c \in MQ} \text{COST}(c) \geq \text{OPT}
\]

for every choice of $T$.

Now, we can conclude the analysis of the lower bound by combing this bound with Claim 5.1, to ensure that
\[
\sum_{c \text{ is } \delta\text{-detectable cell } c \in MQ} \text{COST}(c) \geq (1 - \varrho/3) \cdot \text{OPT}
\]

with probability at least $1 - \frac{2}{3} \varrho$. In summary, we have shown that
\[
\sum_{c \text{ is } \delta\text{-detectable cell } c \in MQ} \text{COST}(c) \leq (1 + \varrho/3) \cdot \text{OPT}
\]

and
\[
\sum_{c \text{ is } \delta\text{-detectable cell } c \in MQ} \text{COST}(c) \geq (1 - \varrho/3) \cdot \text{OPT} ,
\]

with probability at least $1 - \frac{2}{3} \varrho$. Thus, with probability at least $1 - \varrho$, it holds that
\[
(1 - \varrho) \cdot \text{OPT} \leq \log \Delta + 1 \sum_{i=0}^{\log \Delta + 1} \sum_{c \text{ is } \delta\text{-detectable cell } c \in G_i} \text{COST}(c)
\]

and
\[
\log \Delta + 1 \sum_{i=0}^{\log \Delta + 1} \sum_{c \text{ is } \delta\text{-detectable cell } c \in G_i} \text{COST}(c) \leq (1 + \varrho) \cdot \text{OPT} ,
\]

what completes the proof. □

### 5.2 Sampling algorithm in streaming setting

In this section, we will implement a variant of the following sampling algorithm in the streaming setting. The algorithm receives a point set $P$ and a sample size $s$ as input, and it uses an $\alpha$-approximation algorithm $A_{\text{px}}$ for optimal cost of facility location of $P$ with $\alpha = O(1)$. Although the algorithm below does not explicitly compute the partition of the input space as defined in the previous sections, it makes implicit use of it by only accepting $\delta$-detectable cells ($\delta$-detectable cells are a subset of $\Lambda$).

Furthermore, for the analysis of the streaming algorithm we will also need a second variant of the algorithm that accepts every maximal light cell, i.e., every cell of $\Lambda$.

**FL-Sampling ($P, s$)**

- For $i = 0$ to $\log \Delta + 1$ do
  - Sample each cell from grid $G_i$ with probability $\frac{s}{\text{OPT}}$ and let $G_i$ be the resulting set
  - Let $F_i = 0$
  - For each $\delta$-detectable (variant 2: maximal light) cell $c \in G_i$ do
    - Determine a $(1 + \varrho)$-approximation $C$ of the facility location cost of $P \cap c$
    - Let $F_i = F_i + C$
  - Return $F = \frac{\Delta \log \Delta}{s} \sum_{i=0}^{\log \Delta + 1} F_i$

Before we start to explain the necessary modifications to implement this algorithm in the streaming model, we will analyze the quality of the sampling of FL-Sampling ($P, s$).

**Lemma 5.2.** Let $1 > \varrho > 0$ be a constant. Let $0 < \varrho < 1$ and let $c = c(\varrho)$ be an appropriate constant as guaranteed to exist by Lemma 5.1. If $s \geq \frac{2\varrho \cdot c}{\varrho + 1} = O(\log^2 \Delta / (\varrho^3))$, $\delta = c\varrho \cdot \delta$, and if the sampling is done pairwise independently within grid $G_i$ and fully independently for different grids, then with probability at least $1 - \varrho$ the estimate returned by algorithm FL-Sampling ($P, s$) satisfies
\[
|F - \text{OPT}_P| \leq O(\varrho) \cdot \text{OPT}_P .
\]

This holds for both variants of the algorithm.

**Proof.** By our choice of $\delta$ with probability at least $1 - \varrho/2$ the partition satisfies Lemma 5.1. We will use Chebyshev’s inequality. We enumerate the $\delta$-detectable cells from grid $G_i$. Let $c_j$ be the $j$-th such cell in the enumeration. Let $X_j$ be the indicator random variable for the event that $c_j$ is sampled and let $Y_j = X_j \cdot \text{COST}(c_j)$. Since every $\delta$-detectable cell is light, we observe that $Y_j \leq 2\kappa$. We can write
\[
F_i = \frac{A_{\text{px}}}{s} \sum_{j} Y_j ,
\]
and 
\[ \mathbb{E}[F_i] = \sum_{\text{\(\delta\)-detectable cell} \in G_i} \mathbb{C}(\varepsilon) , \]
and
\[ \mathbb{E}[F] = \mathbb{E}[\sum_{i=0}^{\log \Delta + 1} F_i] . \]

By Lemma 5.1 (or, for the second variant, by Theorem 2.1 and the fact that the maximal light cells form a partition of the input space) we also have
\[
(1 - \varepsilon) \cdot \text{OPT}_P \leq \mathbb{E}[F] \leq (1 + \varepsilon) \cdot \text{OPT}_P
\]
with probability at least \(1 - \rho/2\). Furthermore, by pairwise independence of the \(Y_j\)
\[
\text{Var}[F_i] \leq \mathbb{E}[F_i^2] = \frac{\text{Apx}^2}{s^2} \sum_j \mathbb{E}[Y_j^2]
\]
\[
= \frac{\text{Apx}^2}{s^2} \cdot \sum_j \mathbb{E}[(\mathbb{C}(\varepsilon))^2 \cdot X_j^2]
\]
\[
= \frac{\text{Apx}^2}{s^2} \cdot \sum_j \mathbb{E}[(\mathbb{C}(\varepsilon))^2 \cdot X_j]
\]
\[
\leq \frac{\text{Apx}^2}{s^2} \cdot (2 \cdot \kappa) \cdot \sum_j \mathbb{E}[\mathbb{C}(\varepsilon) \cdot X_j]
\]
\[
= \frac{2 \cdot \text{Apx}^2 \cdot \kappa}{s} \cdot \sum_j (\mathbb{C}(\varepsilon) \cdot \mathbb{E}[X_j])
\]
\[
= \frac{2 \cdot \text{Apx} \cdot \kappa}{s} \cdot \mathbb{E}[F_i] ,
\]
and by independence of sampling in different grids
\[
\text{Var}[F] = \sum_{i=0}^{\log \Delta + 1} \text{Var}[F_i]
\]
\[
\leq \frac{2 \cdot \text{Apx} \cdot \kappa}{s} \cdot \sum_{i=0}^{\log \Delta + 1} \mathbb{E}[F_i]
\]
\[
= \frac{2 \cdot \text{Apx} \cdot \kappa}{s} \cdot \mathbb{E}[F]
\]
\[
\leq \frac{2 \cdot (\alpha \cdot \text{OPT}_P) \cdot \kappa}{s} \cdot (1 + \varepsilon) \cdot \text{OPT}_P
\]
\[
= \frac{2 \cdot (1 + \varepsilon) \cdot \alpha \cdot \text{OPT}_P^2 \cdot \kappa}{s} ,
\]
where we use (5.6) in the last inequality. By Chebyshev's inequality and (5.6) we get,
\[
\text{Pr}[|F - \text{OPT}_P| > \varepsilon \cdot \text{OPT}_P]
\]
\[
\leq \frac{2 \cdot (1 + \varepsilon) \cdot \alpha \cdot \text{OPT}_P^2 \cdot \kappa}{4 \cdot \varepsilon^2 \cdot \text{OPT}_P^2}
\]
\[
\leq \frac{\rho}{2}.
\]
for our choice of \(s\). Hence, the lemma follows by the union bound. \(\Box\)

In the streaming setting we will not be able to exactly determine whether a cell is \(\delta\)-detectable or whether it is maximal light. However, we can make sure that we accept all \(\delta\)-detectable cells and do not accept any cell that is not maximal light. The following corollary says that this is sufficient for the purpose of approximation.

**Corollary 5.1.** If instead of accepting \(\delta\)-detectable cells or maximal light cells, algorithm FL-SAMPLING accepts a set of cells which contains all \(\delta\)-detectable cells and which is a subset of maximal light cells, then, under the same premise as in Lemma 5.2, with probability at least \(1 - 2\rho\) the estimator returned by FL-SAMPLING satisfies
\[
|F - \text{OPT}_P| \leq O(\varepsilon) \cdot \text{OPT}_P .
\]

*Proof.* We have shown that the algorithm FL-SAMPLING computes a \((1 + \mathbb{O}(\varepsilon))\)-approximation if it accepts all maximal light cells and it computes one if the algorithm only accepts all \(\delta\)-detectable cells. Let us call all maximal light sample cells \(C\) and all \(\delta\)-detectable sample cells \(D\); let the output value of the algorithm accepting all maximal light cells be \(F_C\) and the output value of the algorithm accepting only all \(\delta\)-detectable cells be \(F_D\). If the algorithm accepts an arbitrary set of cells \(A\) with \(D \subseteq A \subseteq C\), then it outputs a value \(F_A\) with \(F_D \leq F_A \leq F_C\). The result follows by the union bound since \((1 - O(\varepsilon))\text{OPT}_P \leq F_D\) and \(F_C \leq (1 + O(\varepsilon))\text{OPT}_P\). \(\Box\)

We now discuss the necessary modifications required to implement algorithm FL-SAMPLING in the streaming setting. We will only prove that the algorithm succeeds with probability at least \(2/3\). In order to improve this probability to \(1 - \rho\), one can use standard amplification techniques, i.e., run \(O(\log 1/\rho)\) independent instances of the algorithm and return the median of the computed values. We will prove the following main theorem.

**Theorem 5.1.** Algorithm FL-SAMPLING can be implemented in the setting of dynamic geometric data streams. This streaming algorithm uses \((\log \Delta \log(1/\rho)/\varepsilon)^{O(1)}\) space and update time and returns in \((\log \Delta \log(1/\rho)/\varepsilon)^{O(1)}\) space and update time and returns in \((\log \Delta \log(1/\rho)/\varepsilon)^{O(1)}\) time an estimator \(F\) that with probability at least \(1 - \rho\) satisfies
\[
(1 - \varepsilon) \cdot \text{OPT}_P \leq F \leq (1 + \varepsilon) \cdot \text{OPT}_P .
\]

Since we do not know the value of \text{Apx} in advance, we will use one instance for any guess of cost \(2^k \cdot f, 0 \leq k \leq 2 \log \Delta\). This requires \(O(\log \Delta)\) such instances, each of which will succeed with probability greater than or equal to \(5/6\). We will also maintain an instance of the \(O(1)\)-approximation algorithm for the cost of the facility location.
problem by Lammersen and Sohler [18] with success probability at least 5/6. Using the result of this algorithm, we can decide afterwards, which instance of our main algorithm we will use. The probability that the \( O(1) \)-approximation algorithm returns a correct result and that this particular instance runs correctly is at least 2/3. Thus, in the following we may assume that we know \( A_{px} \) and we need to prove that there is an algorithm with success probability at least 5/6.

5.2.1 Sampling in dynamic data streams. Our next change is that we determine \( \mathcal{G}_1 \) to be the set of cells that are mapped to 1 by a pairwise independent hash function \( h_i \) with range \( 1, \ldots, r = \lceil A_{px}/s \rceil \). Now that we have defined the set \( \mathcal{G}_1 \) implicitly, a critical point is that \( \mathcal{G}_1 \) can be very large and so we cannot store all points that are mapped to a cell in \( \mathcal{G}_1 \).

This may be overcome by observing that in expectation \( \mathcal{G}_1 \) contains only a few maximal light cells, and so also a few \( \delta \)-detectable cells. Indeed, since the parent of a maximal light cell in grid \( G_i \) is a heavy cell in grid \( G_{i+1} \), the number of maximal light cells in grid \( G_i \) is bounded by four times the number of heavy cells in grid \( G_{i+1} \). Now, we observe that the number of heavy cells in grid \( G_{i+1} \) is upper bounded by

\[
O(\text{OPT}_P/T) = O(\text{A}_{px}/T),
\]

where \( T \) is the random threshold chosen in our algorithm to define heavy cells. Hence, the expected number of maximal light cells in \( \mathcal{G}_1 \) is \( O(s/T) \).

In the following, it will be helpful to first assume that all cells are either \( \delta \)-detectable or empty. In the whole section we will assume \( \delta = c \varepsilon / \log \Delta \), where \( c = c(\varrho) = c(1/200) \) is the constant from Lemma 5.2. We have seen that \( \mathcal{G}_1 \) in expectation contains only very few \( \delta \)-detectable cells. If we use a second pairwise independent hash function \( h'_i \) to map these cells to a random value in \( \{1, \ldots, t\} \) for sufficiently large \( t = \Omega(\varepsilon^{-2}/T^2) \), then with high probability there will be no collision among them. Our two stage hashing scheme now induces substreams of the input stream. First of all, in grid \( G_i \) we will only consider points inside cells that map to 1 under \( h_i \) (subsampling stage). Then the second stage subdivides our subsampled stream into \( t \) substreams, each of which under our assumption will contain all points of at most one \( \delta \)-detectable cell. We may assume, without loss of generality, that our streams consist only of insertion since all operations we will use are linear and so an insertion followed by a deletion cancel out.

We will now explain how to drop our simplifying assumptions. If more than one cell is mapped to the same bucket, we will treat all points inside these cells as if they come from a single cell. We show that the points that are mapped there from cells other that our \( \delta \)-detectable cell only insignificantly changes the facility location of this cell. For this purpose we proceed as follows. Whenever a point inside a cell \( \epsilon \) in \( \mathcal{G}_1 \) is encountered in the stream, we subtract the coordinates of the lower left corner of its cell to obtain a new point (vector) \( p' \) and then we pass this point to a substream \( h'_i(\epsilon) \). For sufficiently large \( t \) each substream will contain points from at most one maximal light cell and some ‘noise points’ mapped from other cells to it, which will be sufficiently small to approximate the cost of the maximal light cell with factor, say, \( (1 + \delta/100) \).

The scheme described above is a modification of an algorithm introduced by [1]. The idea to use random shifts in this context is from [14]. See also [15].

It remains to explain how to detect whether a substream contains a \( \delta \)-detectable cell. To do that, we determine the points inside the parent cell of each sample cell in \( \mathcal{G}_1 \) and we hash them to different substreams to ensure that with high probability no stream contains more than one heavy cell. We summarize the above discussion in the following algorithm.

**StreamingSampleCell** \((i)\)

- Let \( r = \lceil A_{px}/s \rceil \) and \( t = c^* \left( \varepsilon^4 \log^2 \Delta + \varepsilon^2 \log \Delta \right) \).
- Choose two pairwise independent hash functions \( h_i : \{1, \ldots, |G_i|\} \rightarrow \{1, \ldots, r\} \) and \( h'_i : \{1, \ldots, |G_i|\} \rightarrow \{1, \ldots, t\} \).
- For each point \( p \) in the stream do
  - Determine cells \( c_1 \in G_i \) and \( c_2 \in G_{i+1} \) that contain \( p \)
  - Let \( p = p - c_1 \), where \( c_1 \) is the lower left corner of \( c_1 \)
  - Let \( p' = p - c_2 \), where \( c_2 \) is the lower left corner of \( c_2 \)
  - If \( h_i(c_1) = 1 \) then
    - Insert \( p \) into \( ([T]/f, \delta/100) \)-Coreset \( MED_{\text{light}} \_h_i'(c_1) \)
    - For each subcell \( \epsilon \) of \( c_2 \) with \( h_i(c_1) = 1 \) do
      - Insert \( p' \) into \( ([T]/f, \delta/100) \)-Coreset \( MED_{\text{heavy}} \_h_i'(c_1) \)

We run algorithm **StreamingSampleCell** \((i)\) for each value of \( i \) in \( \{0, 1, \ldots, \log \Delta + 1\} \) to obtain estimates for all \( F_i \). In algorithm **StreamingSampleCell** we use the \((k, \delta)\)-coreset data structures of [8] for the \( k \)-median problem maintained in the streaming fashion. Let \( P \) be a (possibly weighted) point set in \( [\Delta]^2 \) in which \( w_p \) is the weight of a point \( p \in P \). In the \( k \)-median problem we would like to find a set \( C \subseteq \mathbb{R}^2 \) of \( k \) centers such that

\[
\text{COST}(P, C) = \sum_{p \in P} \min_{c \in C} w_p \cdot d(p, c)
\]

is minimized. Then a coreset is defined as follows.

**Definition 5.1.** ([11]) A weighted set \( S \subseteq \mathbb{R}^2 \) is called a \((k, \delta)\)-coreset for \( P \) if for any set \( C \subseteq \mathbb{R}^2 \) with \( |C| \leq k \) we have

\[
|\text{COST}(S, C) - \text{COST}(P, C)| \leq \delta \cdot \text{COST}(P, C).
\]
Observe that a \((k, \delta)\)-coreset is also a \((k', \delta)\)-coreset for any \(k' < k\). In our paper we will use the \((k, \delta)\)-coreset data structure from [8], whose properties are described in the following theorem.

**Theorem 5.2.** ([8]) Let \(P \subseteq [\Delta]^2\) be a point set given in the dynamic geometric data stream model. Let \(k \geq 1\), \(\delta > 0\), and \(0 < \varrho < 1\). There exists a randomized streaming algorithm that with probability \(1 - \varrho\) maintains a \((k, \delta)\)-coreset of \(P\) using the algorithm by Har-Peled and Mazumdar. The streaming algorithm processes \(\mathbf{INSERT}(p)\) and \(\mathbf{DELETE}(p)\) operations in \((k \log \Delta \log(1/\varrho) / \delta)^{O(1)}\) time.

For each \(i \in \{0, 1, \ldots, \log \Delta + 1\}\) and using \(\varrho = 1/(200 \log \Delta)\), we maintain a \(((T/f), \delta/100)\)-coreset \(\mathbf{MED}^{\text{light}}_{i,j}\) for each substream of the points mapped to \(j\) and another \(((T/f), \delta/100)\)-coreset \(\mathbf{MED}^{\text{heavy}}_{i,j}\) to maintain the points mapped to the parent cell of a cell mapped to \(j\). Thus, the coreset data structures are simultaneously correct with probability at least \(99/100\).

**5.3 Extraction algorithm and analysis.** We will now state the main properties of the maintained statistics. We will use \(\mathbf{COST}(\mathbf{MED}^{\text{light}}_{i,j})\) and \(\mathbf{COST}(\mathbf{MED}^{\text{heavy}}_{i,j})\) to refer to the cost of an optimal facility location solution of the coreset returned by the data structure.

**Lemma 5.3.** Let \(C = \{\varepsilon \in \mathcal{S}_1 : \varepsilon \text{ is } \delta\text{-detectable}\}\) and let \(L = \{\varepsilon \in \mathcal{S}_i : \varepsilon \text{ is maximal light}\}\). If \(t \geq c^* \left(\frac{\log^2 \Delta}{T/2} + \frac{s \log^2 \Delta}{T/2}\right)\) for sufficiently large \(c^*\), then with probability at least \(1/(200 \log \Delta)\) the following statements simultaneously hold:

- For every cells \(c_1, c_2 \in L\) we have \(h_i'(c_1) \neq h_i'(c_2)\).

- If \(\mathbf{COST}(\mathbf{MED}^{\text{light}}_{i,j}) \leq (1 - \delta/2) \cdot T\) and \(\mathbf{COST}(\mathbf{MED}^{\text{heavy}}_{i,j}) \geq (1 + \delta/2) \cdot T\) then there is a cell \(\varepsilon \in L\) with \(h_i'(\varepsilon) = j\).

- For every cell \(\varepsilon \in L\) we have \((1 - \delta/100) \cdot \mathbf{COST}(\varepsilon) \leq \mathbf{COST}(\mathbf{MED}^{\text{light}}_{i,j}) \leq (1 + \delta/100) \cdot \mathbf{COST}(\varepsilon) + \frac{\delta}{100} \cdot T\).

- For every parent cell \(\varepsilon'\) of a cell \(\varepsilon \in L\) \((1 - \delta/100) \cdot \mathbf{COST}(\varepsilon') \leq \mathbf{COST}(\mathbf{MED}^{\text{heavy}}_{i,j}) \leq (1 + \delta/100) \cdot \mathbf{COST}(\varepsilon') + \frac{\delta}{100} \cdot T\).

Before we prove Lemma 5.3, we will show how to extract an approximation for the cost of the facility location problem from our statistics.

We will need the following procedure to approximate the cost of the facility location problem, if there is a promise that the cost is not too large. We compute \((1 + \delta/100)\)-approximation of the cost of the \(k\)-median problem from the coresets using the algorithm by Har-Peled and Mazumdar [11] and add the term \(kf\) to obtain the corresponding facility location cost for each choice of \(k \in \{1, \ldots, \lfloor T/f \rfloor\}\). To obtain the cost of the facility location problem we finally choose the cost which is minimum of all facility location costs for all values of \(k \in \{1, \ldots, \lfloor T/f \rfloor\}\). This gives a \((1 + \delta/100)\)-approximation for the facility location cost.

Our next step will be to define the extraction algorithm. Given Lemma 5.3, the algorithm will check for entries in the hash tables such that the cost of \(\mathbf{MED}^{\text{light}}_{i,j}\) is at most \((1 - \delta/2)T\) and the cost of the corresponding \(\mathbf{MED}^{\text{heavy}}_{i,j}\) is at least \((1 + \delta/2)T\). Given the first and second item of Lemma 5.3, we know that every value extracted corresponds to a unique maximal light cell. Furthermore, by the third and fourth item in Lemma 5.3 and the fact that we compute a \((1 + \delta/100)\)-approximation, we know that for every \(\delta\)-detectable cell \(\varepsilon\) with \(h_i(\varepsilon) = 1\) we extract a value into \(F_i\). Thus, our sample set satisfies the requirements of Corollary 5.1. The multiplicative and additive error in the costs of the cells is \(O(\varepsilon \cdot \mathbf{OPT}_P)\) with probability at least \(99/100\). Furthermore, with probability at least \(99/100\) Lemma 5.3 is true for all values of \(i\). Then Corollary 5.1 is satisfied with probability at least \(1 - 2\varrho\), where we use \(\varrho = 1/200\). Hence, the corollary is satisfied with probability at least \(99/100\). The maintained coreset data structures work correctly with probability at least \(99/100\). Thus, by the union bound, the algorithm succeeds with probability at least \(5/6\) as required.

**5.4 Proof of Lemma 5.3.** We will prove our lemma in several steps. We start by defining \(\mathbf{COST}^*(\varepsilon)\) of a cell \(\varepsilon\) to be the cost of an optimal solution, where a single facility is opened in the middle of the cell for free (and where additional facilities may be opened at a cost of \(f = 1\)). We call a cell \textit{big} if it has \(\mathbf{COST}^*(\varepsilon) \geq \lambda T\), for \(\lambda \leq \frac{\delta^2 T}{c'^* \log \Delta}\) for an appropriate constant \(c'^*\). We call a cell \textit{important} if it is either maximal light or big. A cell that is not important will be called \textit{unimportant}.

Our first step in order to prove the lemma will be to show that no important cells collide under \(h'_i\). This implies the first item of Lemma 5.3. In order to do so, we first have to prove a bound on the number of important cells that are in \(\mathcal{S}_i\).
get this bound, we show the following.

**CLAIM 5.2.**
\[
\sum_{c \in G_i : c \text{ big}} \text{COST}^+(c) = O(\text{OPT}_P).
\]

**Proof.** For each big cell \( c \), we open all facilities that are opened in an optimal solution in \( c \) and all of its neighboring cells. Furthermore, we open the free facility in the center of each big cell. The connection cost for every point will be at most that of an optimal solution. The cost for opening the facilities that are opened in an optimal solution is at most 9 times the opening cost of the optimal solution. Hence the claim follows. \( \square \)

The following corollary is immediate.

**COROLLARY 5.2.** The number of big cells in grid \( G_i \) is \( O(\frac{\text{OPT}_P}{\lambda T}) \).

We also observe that the parent cell of every maximal light cell is heavy and so it is big. Since every such cell has at most four maximal light subcells, we obtain the following.

**COROLLARY 5.3.** The number of maximal light cells in grid \( G_i \) is \( O(\frac{\text{OPT}_P}{\lambda T}) \).

This immediately implies the following claim.

**COROLLARY 5.4.** The number of important cells in grid \( G_i \) is \( O(\frac{\text{OPT}_P}{\lambda T}) \).

Now let us define \( \mathcal{I} \) to be the set important cells \( c \) in \( \mathcal{G}_i \).

**CLAIM 5.5.** There exists a constant \( c > 1 \) such that
\[
\Pr \left\{ \sum_{c \in \mathcal{I}} \text{COST}^+(c) \geq \frac{s}{c} \cdot \frac{\lambda \log T}{\lambda T} \right\} \leq \frac{1}{1000 \log \Delta}.
\]

**Proof.** We have \( \mathbb{E} |\mathcal{I}| = O(\frac{\text{OPT}_P}{\lambda T}) \cdot \text{OPT}_P = O(\frac{\text{OPT}_P}{\lambda T}) \). Then the claim follows from Markov’s inequality. \( \square \)

Next we prove our first intermediate result which implies the first item of Lemma 5.3.

**CLAIM 5.6.** There is a constant \( c \) such that for \( t \geq \frac{c s \log \Delta}{\lambda T} \) the following holds:
\[
\Pr \left\{ \exists c_1, c_2 \in \mathcal{I} \text{ with } h'_i(c_1) = h'_i(c_2) \right\} \leq \frac{1}{500 \log \Delta}.
\]

**Proof.** For a moment, let us assume that \( |\mathcal{I}| \leq c' \cdot \frac{s}{\lambda T \log \Delta T} \) for some constant \( c' > 1 \). Then, by pairwise independence of \( h'_i \), the probability that two fixed cells collide is \( 1/t \). Taking the union bound over all pairs gives that with probability at most \( \frac{1}{t} \cdot \left( \frac{c' s}{\lambda T \log \Delta T} \right)^2 \) there is a collision of maximal light cells. For \( t \geq \frac{c s \log \Delta}{\lambda T} \) and \( c \geq 1000(c')^2 \), this probability is at most \( 1/(1000 \log \Delta) \). Due to Claim 5.3, with probability at least \( 1 - 1/(1000 \log \Delta) \), we have \( |\mathcal{I}| \leq c' \cdot \frac{s}{\lambda T \log \Delta T} \), which completes the proof. \( \square \)

It remains to analyze how the “noise” introduced by unimportant cells influences our result. Let us use \( \mathcal{U} \) to denote the unimportant cells in \( \mathcal{G}_i \).

**CLAIM 5.7.** There exists a constant \( c > 1 \) such that
\[
\Pr \left\{ \sum_{c \in \mathcal{U}} \text{COST}^+(c) \geq \frac{s}{c} \cdot \frac{\lambda \log \Delta}{\lambda T} \right\} \leq \frac{1}{1000 \log \Delta}.
\]

**Proof.** Using the same arguments as in the proof of Claim 5.2 we can show that
\[
\sum_{c \in G_i : c \text{ unimportant}} \text{COST}^+(c) = O(\text{OPT}_P).
\]

Since each cell is in \( \mathcal{G}_i \) with probability \( 1/r \), we get
\[
\mathbb{E}\left[ \sum_{c \in \mathcal{U}} \text{COST}^+(c) \right] = O(s). \quad \text{Now, the claim follows by Markov’s inequality.} \quad \square
\]

We condition on the fact that Claim 5.5 is satisfied. We will use Chebyshev’s inequality for fixed \( j \) and then use the union bound. For a cell \( c \) let
\[
X_{c,j} = \begin{cases} 
\text{COST}^+(c) & \text{if } h'_i(c) = j, \\
0 & \text{otherwise}.
\end{cases}
\]

We have \( \mathbb{V}ar[X_{c,j}] \leq \frac{1}{t} (\text{COST}^+(c))^2 \). Let us define
\[
X_j = \sum_{c \in \mathcal{U}} X_{c,j}.
\]

We have
\[
\mathbb{E}[X_j] \leq \frac{cs \log \Delta}{t} \leq \delta T/200,
\]
for our choice of \( t \) and for \( c'' \) being a sufficiently large constant and where \( c \) is the constant from the previous claim. By pairwise independence,
\[
\mathbb{V}ar[X_j] \leq \sum_{c \in \mathcal{U}} \mathbb{V}ar[X_{c,j}] \leq \frac{1}{t} \cdot \sum_{c \in \mathcal{U}} (\text{COST}^+(c))^2 \leq \frac{1}{t} \cdot c \cdot \lambda \cdot T \cdot \left( \sum_{c \in \mathcal{U}} (\text{COST}^+(c)) \right) \leq c \cdot \lambda \cdot T \cdot s,
\]

Copyright © SIAM. Unauthorized reproduction of this article is prohibited.
where $c$ is the constant from the previous claim.

By Chebyshev’s inequality we have

$$\Pr[|X_j - \mathbb{E}[X_j]| \geq \delta \cdot T/200] \leq \frac{40000 \cdot \text{Var}[X_j]}{\delta^2 \cdot T^2} \leq \frac{40000 \cdot \lambda \cdot c \cdot s}{\delta^2 \cdot T \cdot t}.$$  

Thus, for large enough $c'$ and $\lambda \leq \frac{\delta^2 T}{c' \cdot s \cdot \log \Delta}$, we get with probability at least $1 - \frac{1}{1000 \log \Delta}$ for all $j$ with $1 \leq j \leq t$ that $|X_j - \mathbb{E}[X_j]| < \delta T/200$. Since Claim 5.5 is satisfied with probability $1 - \frac{1}{2000 \log \Delta}$, the claim follows. \hfill \Box

It remains to argue why the above claim implies the remaining items of Lemma 5.3. We have proved that the sum of $\text{COST}^*$ of the unimportant cells mapped to a fixed bucket $j$ is at most $\delta T/100$. This implies that any bucket to which no big cells is mapped has a bucket where sum of $\text{COST}$ of the remaining items of Lemma 5.3. We have proved that the probability at least $c \delta/(1 + \lambda) \cdot \lambda \cdot \log \Delta$ for every important cell $i \in \mathcal{I}_1$ we get $((1 - \delta/100) \cdot \text{COST}(i) \leq \text{COST}(Med_{i,h(i)}^{\text{light}})$ and $\text{COST}(Med_{i,h(i)}^{\text{heavy}}) \leq (1 + \delta/100) \cdot \text{COST}(i) + \delta T/100 + 1 \leq (1 + \delta/100) \cdot \text{COST}(i) + \delta T/50$. The same bound holds for big cells $c$ from $G_{i+1}$ and $\text{COST}(Med_{i,h(i)}^{\text{light}}(c))$. This proves items three and four.

In order to prove the second item, let us assume that there is a bucket $j$ that satisfies the second statement. The bounds above imply that there is also a cell $c$ with $h_i(c) = 1$ and whose parent cell is heavy. Furthermore, $c$ cannot be heavy and so it is maximal light, which proves the second item. Hence the lemma follows for $t \geq c^*(1 - \frac{\log^2 \Delta}{T^2})$ for sufficiently large $c^*$.

References


